APPLICATION OF AN OPERATOR SPLITTING ALGORITHM FOR COVECTION DIFFUSION PROBLEM

by

K. M. PILLAI

DEPARTMENT OF MECHANICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

APRIL, 1991

APPLICATION OF AN OPERATOR SPLITTING ALGORITHM FOR COVECTION DIFFUSION PROBLEM

A Thesis Submitted
in Partial Fulfilment of the Requirements
for the Degree of

MASTER OF TECHNOLOGY

by

K. M. PILLAI

to the

DEPARTMENT OF MECHANICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KANPUR
APRIL, 1991

ME-1991-M-PIL-APP.

CERTIFICATE

This M. Tech. thesis entitled, "Application of an Operator Splitting Algorithm for Convection-diffusion Problems" by Mr. K.M. Pillai, Roll No. 8910515 has been done under our supervision and has not been submitted elsewhere for a degree.

K Muralidhar

(Dr.K. Muralidhar)
Assistant Professor
Indian Institute of Technology Kanpur

(Dr. P.S. Ghoshdastidar)
Assistant Professor

April, 1991

TABLE OF CONTENTS

ABSTRACT

A novel numerical technique of solving convection-diffusion type equations, called the operator splitting technique has been used in the present study to solve model one and two dimensional heat transfer problems. Results of this study have been compared to those of two other conventional differencing schemes namely the upwind technique and the central difference technique. These results are further compared with the analytical solution wherever possible. It is found that for low Peclet number flows all the three schemes give results identical to the analytical solutions. But at high Peclet numbers, convection dominates over diffusion and upwinding shows significant errors. Central differencing is also found to be unsatisfactory. However, the operator splitting solution comes close to the analytical solution. Hence we conclude that this scheme is relatively free from the maladies afflicting the conventional lower order finite difference schemes for a wide range of Peclet numbers.

In the second part of the thesis, two phase oil-water flow through the porous media is numerically studied. In this problem, the operator splitting algorithm is used to solve the energy equation for temperature. The relative merits of cold and hot water injection in oil recovery have been studied here. In these problems velocity is not prescribed independently and must be obtained from the simultaneous solutions of pressure and temperature equations. The following conclusions have been

arrived at in this study.

i) Raising the formation temperature (T_f) improves the oil recovery.

ii) Hot water injection is better than the cold water injection in terms of oil recovery.

iii) Loss of heat to the surrounding rocks adversely affect the oil production.

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDICES

NOMENCLATURE

This terminology is applicable everywhere in this work except in Chapter 4 and Appendices B(b), E, F and G. Nomenclature of these sections is given in Chapter 4, Section 4.3.

Symbols

x,y	Global ordinate and abscissa
Δx, Δy	Grid size along x and y axes respectively
u,v	Fluid velocities along x and y axes respectively
N,M	Total number of nodes along x and y axes respectively
X,Y	Domain dimensions along x and y axes respectively
t	Time
Δt	Size of a time step
Pe	Peclet number
Pe_g	Grid Peclet number
η, ξ	Coordinates parallel and normal to the mean flow respectively

Superscript

p	Present time step
$\sim$	Vector form

Subscript

i,j	Nodal indices along x and y axes respectively.

ACKNOWLEDGEMENTS

The author would like to express his deep gratitude and sincere appreciation to Dr. K. Muralidhar and Dr. P.S. Ghoshdastidar under whose able guidance this work was done.

The author is thankful to Mr. U.S. Misra who typed and printed the manuscript. He is also grateful to Mr. S.S. Kushwaha and Mr. G.K. Shukla (of Mechanical Engineering Design Cell) who drew the graphs and the figures.

K.M. Pillai

CHAPTER 1

INTRODUCTION

1.1 Introduction

Heat and mass transfer, fluid flow, chemical reaction and other related processes occur in engineering equipment, in the natural environment, and in living organisms. That these processes play a vital role can be observed in a great variety of practical situations. Nearly all methods of power production involve fluid flow and heat transfer as essential processes. The same processes govern the heating and airconditioning of the buildings. Major segments of the chemical and metallurgical industries use components such as furnaces, heat exchangers, condensers, and reactors, where thermofluid processes are at work. Forces that sustain motion of aircraft and rockets arise from fluid flow, heat transfer, and chemical reaction. In the design of electrical machinery and electronic circuits, heat transfer is often a limiting factor. The spread of pollution of the natural environment is governed by heat and mass transfer principles. Hence there is a need to model and understand the processes of heat and mass transfer in the greatest possible detail.

The prediction of performance of a given physical system consists of finding the values of the relevant variables such as velocity, pressure, temperature and concentration of the relevant chemical species in the processes of interest at various instants of time. The predictions should also describe the effect of

changes in geometry, flow rates, temperature and fluid properties on the performance of the system. Heat transfer and fluid flow processes can be studied either from laboratory experiments or by theoretical modelling. Experimental studies involve tests conducted on a full scale or scaled down model of the physical system. However, these tests are in most cases expensive and time consuming. Further the small scale models do not always simulate all the features of the full scale physical situation; frequently, important features such as combustion or boiling are omitted from the model tests.

A theoretical calculation uses a mathematical model consisting of a set of differential equations that describe the physical system at every point inside the material domain and at every point in time. Very often, an analytical solution of these equations for a practical problem can not be found, thereby limiting its use for an engineer. Fortunately, the development of grid based numerical methods such as the finite element method and the finite difference method and the spectacular progress in recent years in the speed of digital computers hold the promise that the mathematical model of almost any practical problem can be worked out. This new methodology for attacking the complex problems in fluid mechanics and heat transfer has become known as Computational Fluid Dynamics (CFD). The main advantage of such a theoretical approach are low cost, remarkable speed with which the numerical investigation can be performed, complete information of all relevant variables and flexibility to simulate realistic conditions of our choice.

Convection-diffusion equations, arising in heat and mass transfer problems, is an important class of problems that is being addressed to in CFD. They make appearance in many diverse forms such as the Navier-Stokes equations, Vorticity Transport equations, Energy Balance equations and the Chemical Transport equation. In general, the transport equations for the velocity components, diffusion coefficients and various scalars like vorticity, mass fraction and specific enthalpy in a general unsteady multidimensional flow are examples of convection-diffusion type equations. There are many numerical techniques such as upwinding, central difference method and hybrid differencing which are available to solve such equations. Of all these techniques, upwinding (or upstream differencing) has become popular.

It is well known that the application of the first order upwind difference scheme to the convection term of the convection-diffusion equation stabilizes computations but introduces errors known as numerical diffusion. Owing to this error, the solution of the equation shows larger presence of diffusion than what is present in the flow. In processes where the fluid velocity is high, convection dominates diffusion and front like solutions can be produced. In such problems, tracking of the fronts is important. Application of upwind technique in these problems introduces such a large artificial diffusion error that it weakens the effects of convection and leads to the smearing of sharp fronts. Fronts can form in certain two phase flow problems as well, even when the average velocity involved is

small. Resolution of these fronts by upwinding can lead to substantial errors unless a highly refined grid is used.

An example of formation of fronts is seen when the hot water injection technique is used to extract oil from an underground oil reservoir. The hot water passes on the thermal energy to oil, thereby reducing its viscosity and improving its mobility. Water does not mix with oil and the former acts as a piston that displaces oil trapped in the pores of rocks. This leads to a moving interface or front between oil and water. It is important that the numerical simulator of the process correctly predicts the front movement as the breakdown of front will severely affect the theoretical recovery efficiency. The mathematical model for the displacement process consists of a system of non-linear partial differential equations whose solutions exhibit moving fronts. The use of upwind differencing has retarded research in this area due to a host of problems. There are the fudging of the fronts, unreliable value of variables and excessive computation costs of the higher order methods that could substitute upwinding.

There are several other examples where the use of upwind technique has not yielded satisfactory results. In a variety of engineering systems, flows take place in a complex geometry, as in diffusers, nozzles and in corrugated channels, or occur in a manner leading to a complex flow field (as in flow behind a cylinder). Conventional numerical solution of such problems using upwind schemes grossly average out the flow structure even at a moderate Reynolds number, say Re ~ 100. This leads to an

under prediction of pressure drops in such devices.

Several efforts have been made so far by different investigators to come up with a method to reduce upwind errors, though only with limited success. This work attempts to find an alternative to upwinding itself. A method called the operator splitting technique, which completely eliminates the need for using the upwind principle, has been used here. Preliminary results presented in this thesis show that this new method is well suited for both convection and diffusion dominated problems.

1.2 Literature Survey

A review of literature shows that the classical first order upwind scheme was first used by Richard Courant. It has been explained in detail by Roache [14]. Spalding [13] proposed a hybrid scheme as an improvement over the upwind scheme. Patankar [13] has recommended a power-law scheme for convective terms in flow and heat transfer problems. Hybrid and power law schemes derive their inspiration from the exponential solution of one dimensional steady flow. This scheme sets the diffusion effect equal to zero at much larger grid Peclet number ($Pe_g > 2$) and therefore is reduced to the upwind scheme. Leonard [1], [2] has strongly criticized the upwind scheme and its derivatives as being stable but very inaccurate because of artificial diffusion. He has commented that the conclusions of the steady one dimensional problem used by upwind, hybrid and power law schemes cannot be generalized to more complex problems involving convection and any combination of streamwise diffusion, cross

stream transport in two dimensions, unsteady boundary conditions and steady or unsteady source terms. Leonard shows that grid refinement can alleviate the problem of artificial diffusion but the necessary degree of refinement is often impractical for engineering applications. His methods (QUICK and QUICKEST) [2] solve the differential equation using a third order upwind method. Though his method is capable of handling large grid Peclet numbers, it is only conditionally stable. In one of his latest papers [11], Leonard discussed higher order schemes such as ULTRT-SHARP which is an alternative for high resolution non-oscillatory multidimensional steady state high speed convective modelling. Murphy [7] applied C^1 cubic Hermite polynomials embedded in an orthogonal collocation scheme to the spatial discretization of the unsteady nonlinear Burger equation as a model of the equation of fluid mechanics. A new finite difference scheme of 2^{nd} order accuracy free of artificial diffusion for solving the steady state incompressible Navier-Stokes equations is presented by Mer et al. [8]. The scheme uses a false transient approach with a combination of variable time step size and over-relaxation for the convection diffusion terms. Lai et al. [9] have developed a second order upwind differencing method for convection-diffusion type equations in porous media. The method utilizes explicit monotonized upwind/central differencing and operator splitting.

The operator splitting technique, which is being presented here as an alternative to the upwind method was first proposed by Yanenko [15]. Various possible types of operator splittings

feasible in Gas dynamics equations have been described in this work. More recently, Issa [3] describes a noniterative method for handling the coupling of the implicitly discretized time dependent fluid flow equations. The method is based on the use of pressure and velocity as dependent variables and is hence applicable to both the compressible and incompressible versions of the transport equations. The main feature of the technique is the splitting of the solution process into a series of steps whereby operations on pressures are decoupled from those on velocity at each step, with split sets of equations being amenable to solution by standard techniques. Cooke [4] has mathematically established that a certain operator splitting of the two dimensional, conservative form, Navier-Stokes equations is second order accurate not only for linear systems, but also is true in the presence of non-linearity. An approximate (linearised) Riemann solver is presented for the solution of the Euler equations of gas dynamics in three dimensions by Glaister [5]. The scheme incorporates operator splitting and is applied to the problem of Mach 3 flow past a forward facing step for some specimen equations of state. Ding and Liu [6] describe a new operator splitting algorithm for the two-dimensional convection-dispersion-reaction equation. The governing equation is split into three successive initial value problems: a pure convection problem, a pure dispersion problem and a pure reaction problem. For the pure convection problem, solutions are found by the method of characteristics. For the pure dispersion problem, time-explicit finite element algorithm is employed. Analytical

solutions are obtained for the pure reaction problem. Results are presented mainly for one dimensional problems.

1.3 Scope of the Present Work

In the first part of this work, simple one and two dimensional test problems have been solved using the operator splitting technique and its results are compared with those of upwind and central difference methods and analytical solution wherever possible.

The test problems considered are as follows:

i) One dimensional transient convection diffusion problem.

ii) Two dimensional transient convection diffusion problem in a rectangular domain with a unit velocity along x axis. A temperature step is given at the inlet and the transient development of mixing layer is predicted.

iii) Two dimensional steady convection diffusion problem with a unit velocity at 45^{o} to the Cartesian grid. As in (ii), two streams of different temperatures flow along the diagonal on its either side. The spatial development of mixing layer is predicted here.

The second part of the thesis concerns the problem of oil recovery by the hot water injection method. The mathematical modelling of the problem consists of three highly non-linear coupled partial differential equations, namely two pressure and one energy equation. The pressure equations are solved simultaneously using finite differences. Fluid velocities are computed from the pressure field, which are later used to compute

the coefficients of energy equation. The energy equation is solved using the operator splitting technique. Results have been presented to understand the following:

(a) Effect of the formation (matrix) temperature in the isothermal (cold water) case.

(b) Comparison of the hot water and the cold water injection techniques.

(c) Effect of heat loss to the surrounding rocks on the hot water injection efficiency.

CHAPTER 2

DESCRIPTION OF THE OPERATOR SPLITTING ALGORITHM AND THE ONE DIMENSIONAL PROBLEM

2.1.1 Description of the Operator Splitting Algorithm

The operator splitting algorithm (to be called OS) is a special case of splitting methods or fractional step methods [15] which reduces the solution of a complicated problem to a successive solution of simpler problems. Various kinds of splittings are possible. These are:

(a) Geometrical splitting which reduces a multidimensional problem to a temporal sequence of problems of smaller dimensions or, in particular to transient one dimensional problems. In the latter case we shall speak of splitting in terms of spatial direction. A.D.I. (alternating direction implicit) scheme for the solution of the transient two dimensional heat equation is an example of this kind.

(b) Physical splitting in which the original physical process is represented as a temporal sequence of processes possessing simpler physical structures. This kind of splitting can be treated as splitting in terms of physical processes.

(c) Analytical splitting which allows various analytical problems to be solved in fractional steps. For example, the restoration of divergence in predictor-corrector schemes or treating the algebraic terms of the original

equations as a separate operator in curvilinear coordinates.

The O.S. algorithm belongs to the splitting of type (C). This algorithm properly identifies the mixed mathematical character of the governing differential equation and solves each homogeneous component of the equation as accurately as possible. Hence the OS algorithm solves the governing equation true to its character.

A typical convection-diffusion equation arising in heat transfer problems is of the form,

$$\frac{\partial T}{\partial t} + (\tilde{u} \cdot \tilde{\nabla})\, T = \nabla^2 T \qquad (2.1)$$

In Cartesian coordinates, $\tilde{\nabla} = (\frac{\partial}{\partial x}, \frac{\partial}{\partial y})$. The mathematical character of the equation is elliptic if $t \longrightarrow \infty$, parabolic if $\tilde{u}$ is small and hyperbolic if $\tilde{u}$ is large in magnitude. In all other cases, the equation is said to be mixed in character. These special cases are summarized below:

Steady state ($t \longrightarrow \infty$) $(\tilde{u} \cdot \tilde{\nabla})T = \alpha \nabla^2 T$ (elliptic) (2.2)

Conduction limit ($|\tilde{u}| \longrightarrow 0$) $T_t = \alpha \nabla^2 T$ (parabolic) (2.3)

Convection limit ($|\tilde{u}| \longrightarrow \infty$) $T_t + \tilde{u} \cdot \tilde{\nabla} T = 0$ (hyperbolic) (2.4)

The OS algorithm treats this problem as follows. At any time level t, solve Equation (2.4) over a time step Δt. This is considered as a predictor step. The corrector step (Equation (2.3)) is solved over the same time interval Δt. At high values of velocity, the correction required is quite small and at low velocities, the predictor step acting alone is inadequate.

However, the two steps combined will furnish the complete solution of the Equation (2.1) irrespective of the magnitude of $\tilde{u}$.

2.1.2 Remarks

1. The above approach is completely free of upwinding and hence devoid of false diffusion errors that usually occur at high Peclet numbers.

2. THe only error arising in the O.S. algorithm is due to time discretization and it is of order Δt. If either the predictor or the corrector equation is solved numerically on a grid of size Δx, then additional truncation errors ($\sim \Delta x^n$, $n \geq 1$) are possible. Hence the overall error is,

$$e \sim O(\Delta x^n, \Delta t), \quad n \geq 1$$

where n depends on the scheme used to solve the diffusive corrector step. Numerical techniques are well established for solving the diffusion equation (2.3). This equation does not have an analytical solution when solved in a complex geometry. Hence for generality a second order finite difference scheme has been used to solve for the corrector step. The discretization error is then given as,

$$e \sim O(\Delta t, (\Delta x)^2)$$

3. The main source of error in solving heat transfer problems occur while approximating the hyperbolic terms $(\tilde{u}.\tilde{\nabla}T)$. In the O.S. algorithm, these terms are isolated from the complete equation and solved as accurately as possible. In this thesis, this part of the problem is solved analytically on a local grid.

4.On non-dimensionalization, the heat transfer equation becomes

$$T_t + (\tilde{u}.\tilde{\nabla})T = \frac{1}{Pe} \nabla^2 T \qquad (2.5)$$

where $Pe = UL/\alpha$ is called as Peclet number (See Appendix B(a)). Pe is a measure of the relative importance of convective transport with respect to diffusive (or conductive) transport. Most of the existing algorithms fail or become inaccurate when Pe is raised. The OS algorithm proposed here is uniformly valid over any range of Pe. Its application to engineering problems where Pe can be small or large forms the focus of this thesis.

As the Peclet number is raised, a discontinuity in the initial conditions will propagate through the flow domain unchanged. This is called a front. Predicting the occurrence of a front and its location is a challanging numerical problem. Fronts can also form in non-linear problems at low or moderate Peclet numbers. The oil recovery problem studied in this thesis is an example of this (See Appendix B (b)).

5. The OS algorithm is only conditionally stable since it involves explicit time marching and is limited by the Courant stability principle. The stability criterion is stated in words as,

"the numerical movement of the front should not exceed one grid spacing in one time increment since the maximum front speed is the fluid velocity".

Hence, we require $\frac{\Delta t}{\Delta x} \leq 1$, for stability.

(In dimensional form, the above condition is, $\frac{u\Delta t}{\Delta x} \leq 1$)

See Appendix C for the derivation of stability criteria using von-Neumann analysis.

2.2.1 The One Dimensional Problem

The application of the OS algorithm for a one dimensional problem is given below.

The governing differential equation is

$$T_t + T_x = \frac{1}{Pe} T_{xx} \qquad (2.6)$$

subjected to initial condition: t = 0, T = 0 and the boundary conditions:

$$x = 0, \quad T = 1$$

$$x \longrightarrow \infty, \; T \longrightarrow 0.$$

This represents a physical problem in which a hot fluid with unit velocity is introduced suddenly (at t = 0) at the inlet of a flow domain (See Figure2.1). The fluid in the physical domain ($0 < x < \infty$) is initially at a constant cold temperature. If the Peclet number is large, a thermal front whose thickness is small will move through the flow region with unit velocity. As $Pe \longrightarrow 0$, the front will be diffuse and the energy transport is due to the molecular action of thermal diffusivity.

2.2.2 Analytical Solution

The closed-form solution of the above equation is,

$$T(x,t) = \frac{2}{\sqrt{\pi}} \int_{\sqrt{\frac{4t}{Pe}}}^{\infty} e^{-\eta^2} e^{Pe\left(\frac{x}{2} - \left(x^2/16\eta^2\right)Pe\right)} \, d\eta$$

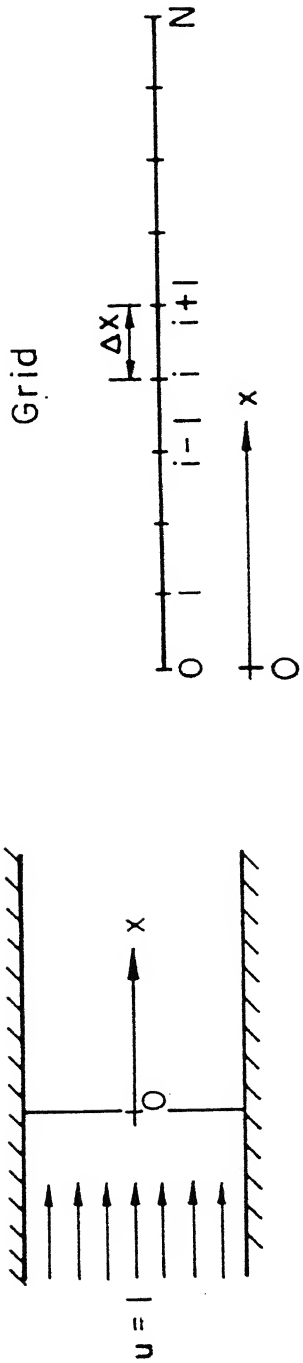


Figure 2.1 : Description of the one dimensional problem.

The above integral is well behaved and is evaluated using Simpson's rule for numerical integration. Values of the upper limit between 8 and 10 reduce the integrand to an order 10^{-6} for any Peclet number and x studied here.

2.2.3 Central Difference Formulation

The governing differential equation is discretized by a fully implicit method in time as,

$$\frac{T_i^{p+1} - T_i^p}{\Delta t} + \frac{(T_{i+1} - T_{i-1})^{p+1}}{2\Delta x} = \frac{1}{Pe}\left(\frac{T_{i+1} + T_{i-1} - 2T_i}{(\Delta x)^2}\right)^{p+1}$$

The above equation determines the value of temperature at the i^{th} node. (See Fig. 2.1). It can be rewritten as,

$$A\ T_{i-1}^{p+1} + B\ T_i^{p+1} + C\ T_{i+1}^{p+1} = D\ T_i^p$$

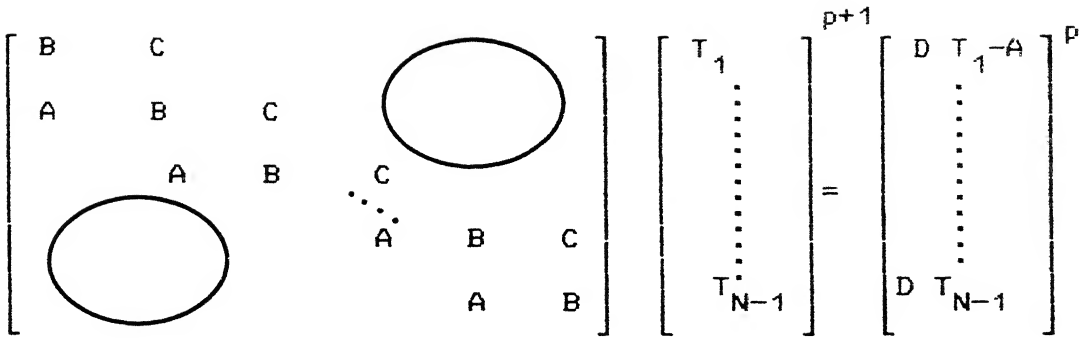
where

$$A = -\frac{1}{2\Delta x} - \frac{1}{Pe(\Delta x)^2}$$

$$B = \frac{1}{\Delta t} + \frac{2}{Pe(\Delta x)^2}$$

$$C = \frac{1}{2\Delta x} - \frac{1}{Pe(\Delta x)^2} \quad \text{and} \quad D = \frac{1}{\Delta t}$$

For a grid system shown in Fig. 2.1, the matrix structure arising froam the central difference/implicit formulation given above is as follows:



The node N is located far enough for the condition $T_N = 0$ to hold. The above matrix is inverted using a tridiagonal algorithm (TDMA).

2.2.4 Upwind Formulation

To generate diagonal dominance, it is essential to use the upwind scheme for the convection term, T_x. This is as follows.

$$u\,T_x = \frac{u_e\,T_i - u_w\,T_{i-1}}{\Delta x}, \quad u > 0$$

$$= \frac{u_e\,T_{i+1} - u_w\,T_i}{\Delta x}, \quad u < 0$$

Subscripts 'e' and 'w' denote the walls of the control volume which are to the east and to the west of the node i respectively. In the present problem u = 1 (>0) and therefore the finite difference formulation becomes,

$$\frac{T_i^{p+1} - T_i^p}{\Delta t} + \frac{T_i^{p+1} - T_{i-1}^{p+1}}{\Delta x} = \frac{1}{Pe}\left(\frac{T_{i+1} + T_{i+1} - 2T_i}{(\Delta x)^2}\right)^{p+1}$$

We write this equation as,

$$\left\{ A\,T_{i-1} + B\,T_i + C\,T_{i+1} \right\}^{p+1} = D\,T_i^p \quad \text{for each node i and}$$

invert the matrix using TDMA. Here,

$$A = -\frac{1}{\Delta x} - \frac{1}{Pe(\Delta x)^2}$$

$$B = \frac{1}{\Delta t} + \frac{1}{\Delta x} + \frac{2}{Pe(\Delta x)^2}$$

$$C = -\frac{1}{Pe(\Delta x)^2} \quad \text{and} \quad D = \frac{1}{\Delta t}$$

Since $|B| > |A| + |C|$, the matrix generated by upwinding has strict diagonal dominance when Dirichlet conditions are prescribed at $x = 0$ and $x \longrightarrow \infty$. The inequality becomes stronger as Δt is reduced.

2.2.5 Operator Splitting Algorithm

The Equation (2.6) is split as,

Predictor: $T_t + T_x = 0$ (2.7)

and

Corrector: $T_t = \frac{1}{Pe} T_{xx}$ (2.8)

with each equation applied over the same interval Δt. The boundary conditions are relevant only over the second corrector step. The convective step Equation (2.7) has an analytical solution,

$T(t-x)$ = constant. This can be written as,

$$T(t + \Delta t - x) = T(t - (x - \Delta t))$$

Choosing $\Delta t = \Delta x$, this becomes,

$$T^{p+1}((t + \Delta t) - x) = T^{p}(t - (x - \Delta x))$$

Hence, the temperature at a point at the new time step (p+1)is obtained by transferring the value prevailing at the current time (p) at the rearward node. If Δt and Δx do not match, then the value of T in the interval (i-1, i) at a distance Δt from node i is transferred to i.

The corrector step is solved by finite differences using an implicit time marching scheme. Hence,

$$\frac{T_i^{p+1} - T_i^p}{\Delta t} = \frac{1}{Pe} \left(\frac{T_{i+1} + T_{i-1} - 2T_i}{(\Delta x)^2} \right)^{p+1}$$

or

$$(A\, T_{i+1} + B\, T_i + C\, T_{i-1})^{p+1} = D\, T_i$$

where

$$A = -\frac{1}{Pe(\Delta x)^2}, \quad B = \frac{1}{\Delta t} + \frac{2}{Pe(\Delta x)^2}, \quad C = -\frac{1}{Pe(\Delta x)^2}, \quad D = \frac{1}{\Delta t}$$

Once again TDMA is used to invert the matrix arising from the above equation. Successive applications of the two steps completes the calculation for a single time increment Δt.

2.2.6 Results

Figures 2.2, 2.3, 2.4 and 2.5 show the plot of temperature distribution as a function of distance for various time levels and Peclet numbers. Typical values of Δx and Δt used in these calculations are 0.1 and 0.1 respectively. All four approaches given above are compared in these figures. The agreement among the numerical predictions and the analytical result is very good at low Peclet numbers ($Pe \leq 10$). At Pe = 100, only the operator

splitting algorithm matches the analytical result closely. The upwind method fails due to false diffusion inherent in its formulation. The central difference method fails due to matrix ill conditioning and loss of diagonal dominance at high Peclet numbers. The operator splitting algorithm is seen to be uniformly accurate at low as well as high Peclet numbers.

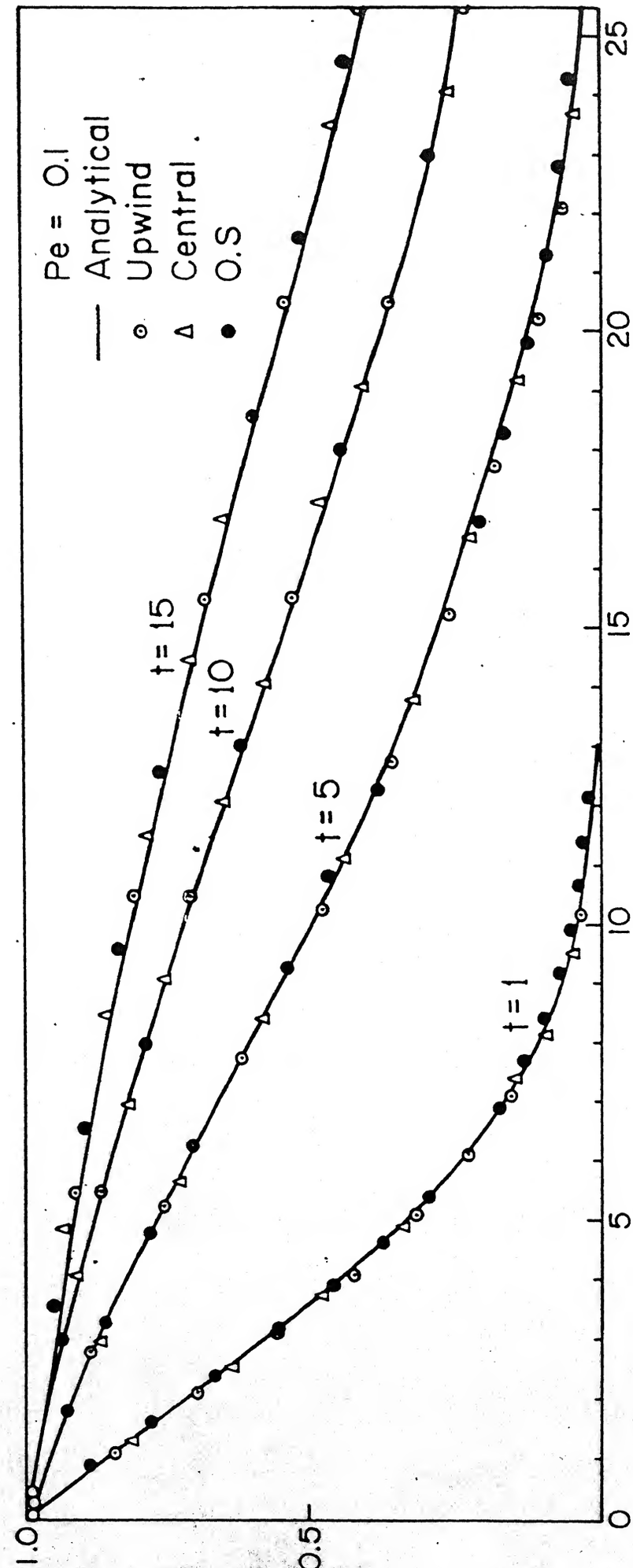
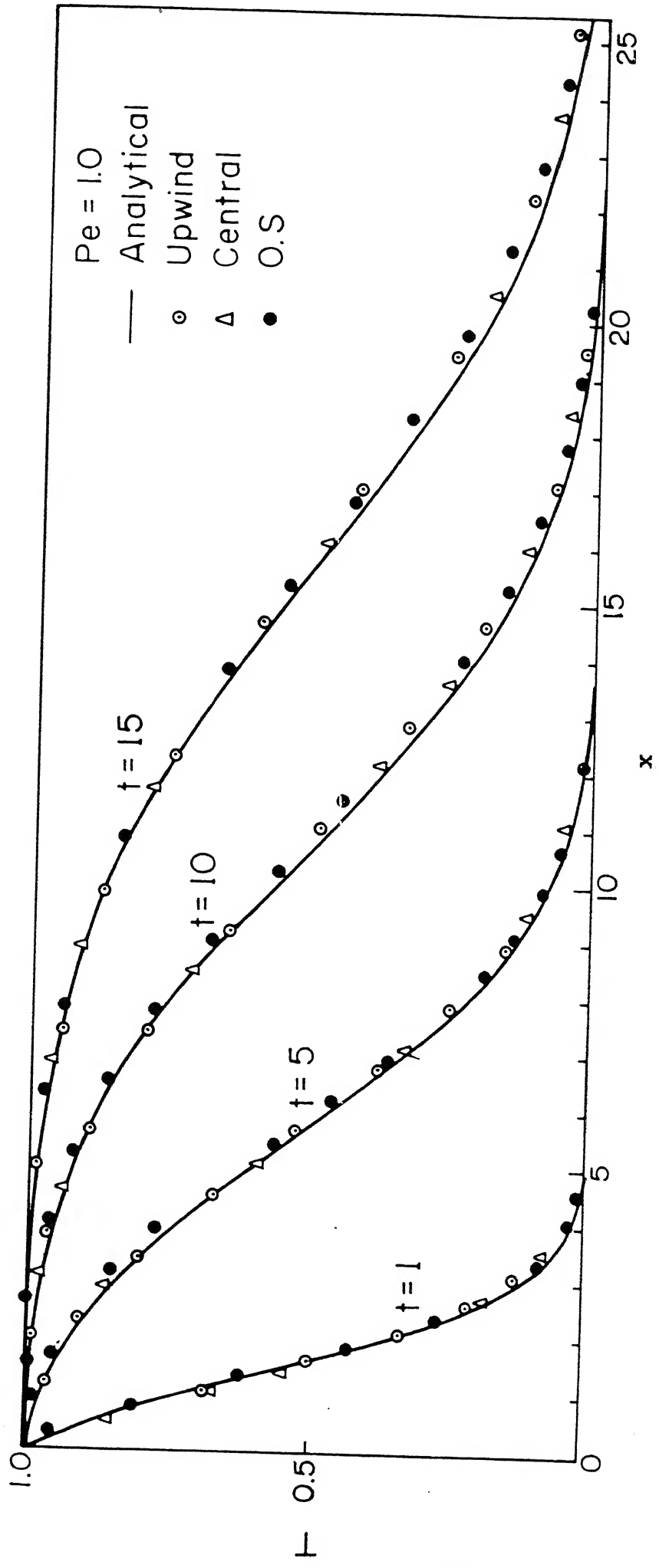


Figure 2.2 : One dimensional problem : Pe = 0.1 .


Pe = 1.0
Analytical
Upwind
Central
O.S
t = 15
t = 10
t = 5
t = 1
T
x
1.0
0.5
0
5
10
15
20
25

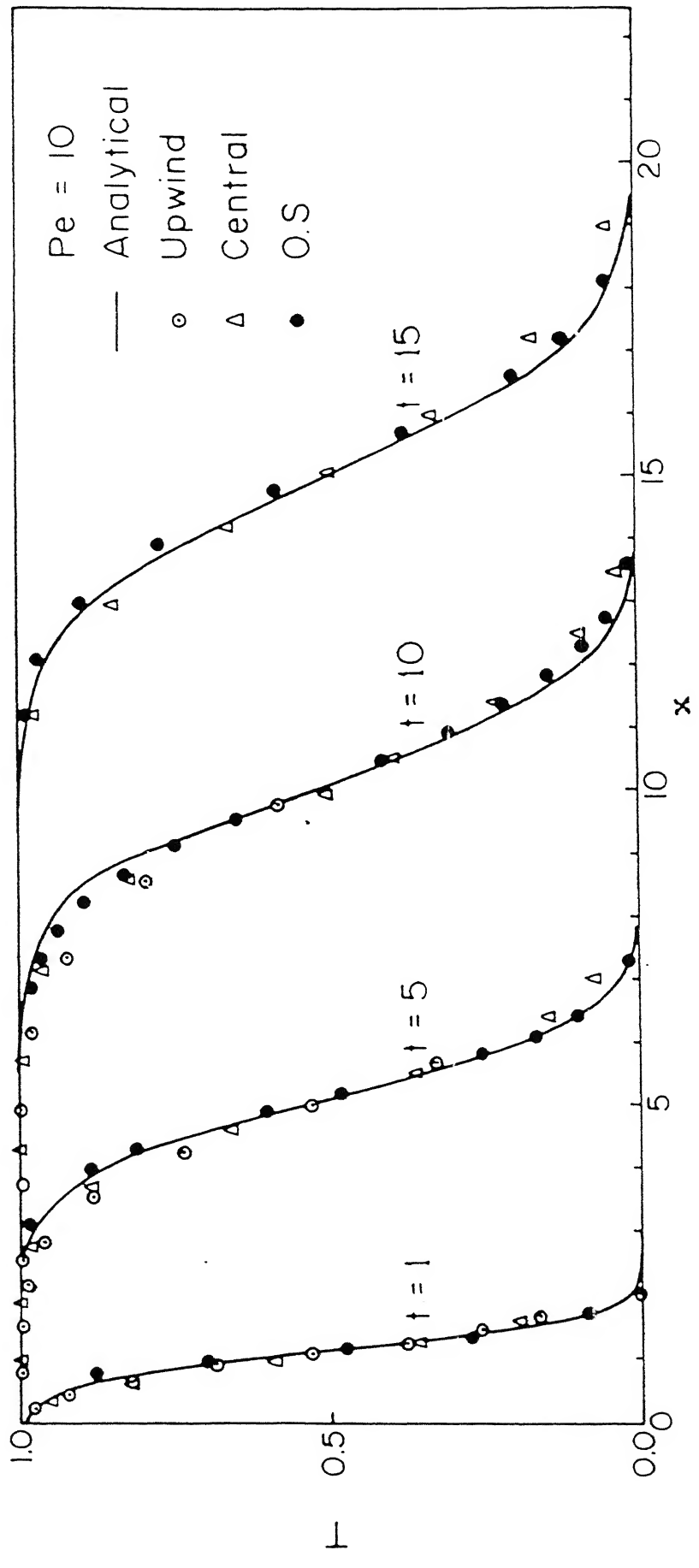


Figure 2.4 : One dimensional problem : Pe = 10.0 .

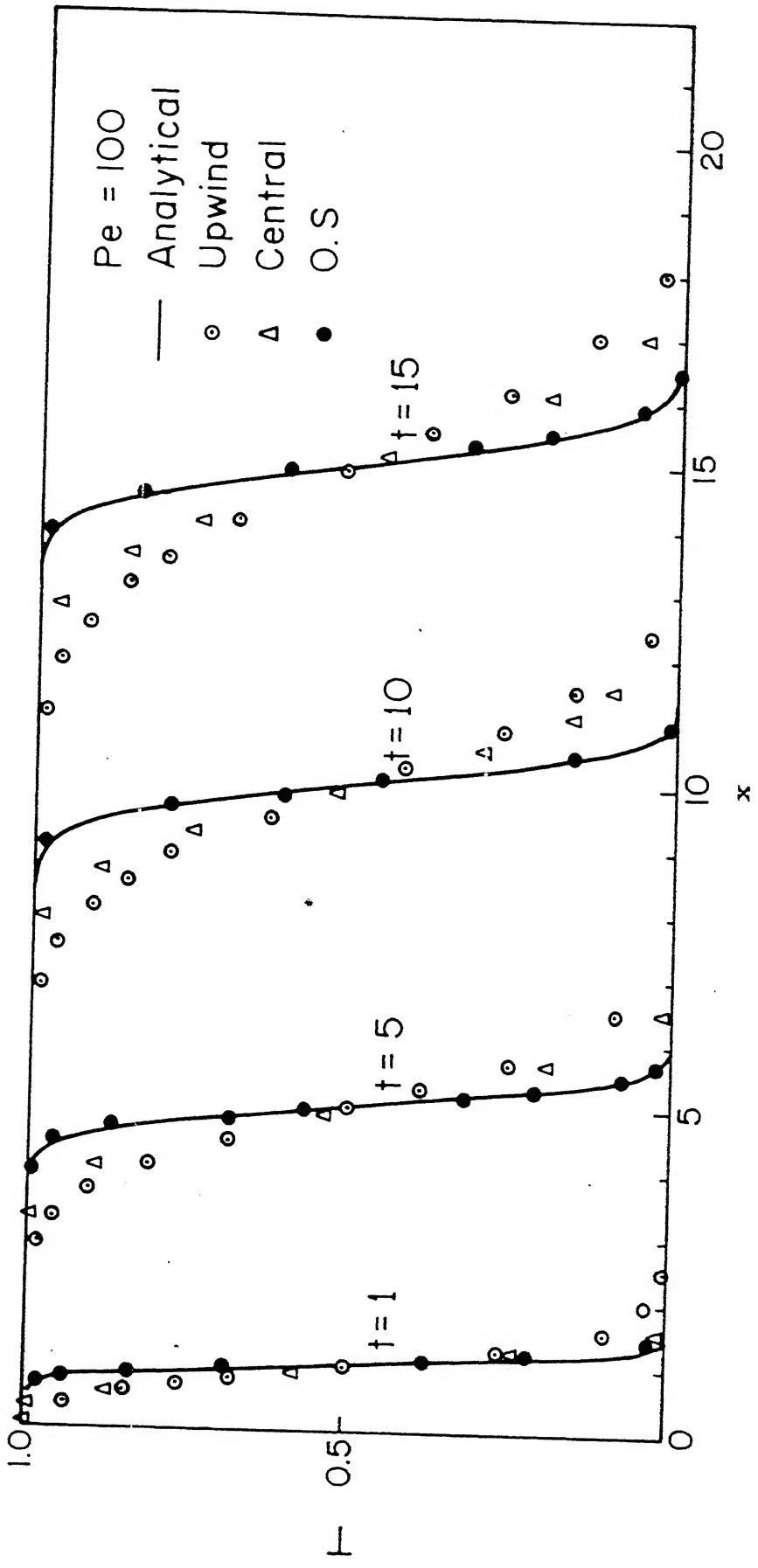


Figure 2.5 : One dimensional problem : Pe = 100.0 .

CHAPTER 3

STUDY OF THE TWO DIMENSIONAL PROBLEM

3.1.1 Description

We consider a two dimensional convective heat transfer problem in which the flow is given as uniform. The governing differential equation for this problem is,

$$T_t + T_x = \frac{1}{Pe} (T_{xx} + T_{yy}) \qquad (3.1)$$

with the boundary conditions

$0 \leq x \leq X$, $y = 0$, $T = 0$;

$0 \leq x \leq X$, $y = Y$, $T = 1$;

$x = 0$, $0 \leq y \leq Y/2$, $T = 0$;

$x = 0$, $Y/2 \leq y \leq Y$, $T = 1$;

$x = X$, $0 \leq y \leq Y$, $\frac{\partial T}{\partial x} = 0$

and the initial condition

$0 < x \leq X$, $0 \leq y \leq Y$, $T \equiv 0$.

See Figure 3.1 for the description of this problem and the finite difference grid. Two parallel streams of equal unit velocity ($u \equiv 1, v \equiv 0$) but unequal temperatures (T=0 and T=1) are taken to come in contact suddenly at the inlet of the rectangular domain. Simultaneously, the upper wall ($y = Y$) is maintained at a temperature T=1. The fluid in the physical domain is initially at a constant cold temperature T=0, possessing the aforementioned unit x-direction velocity. A mixing layer will form at the interface of the two streams in which the temperature gradually changes from the higher value to the lower one. The thickness of this layer grows in the downstream direction. At high Peclet

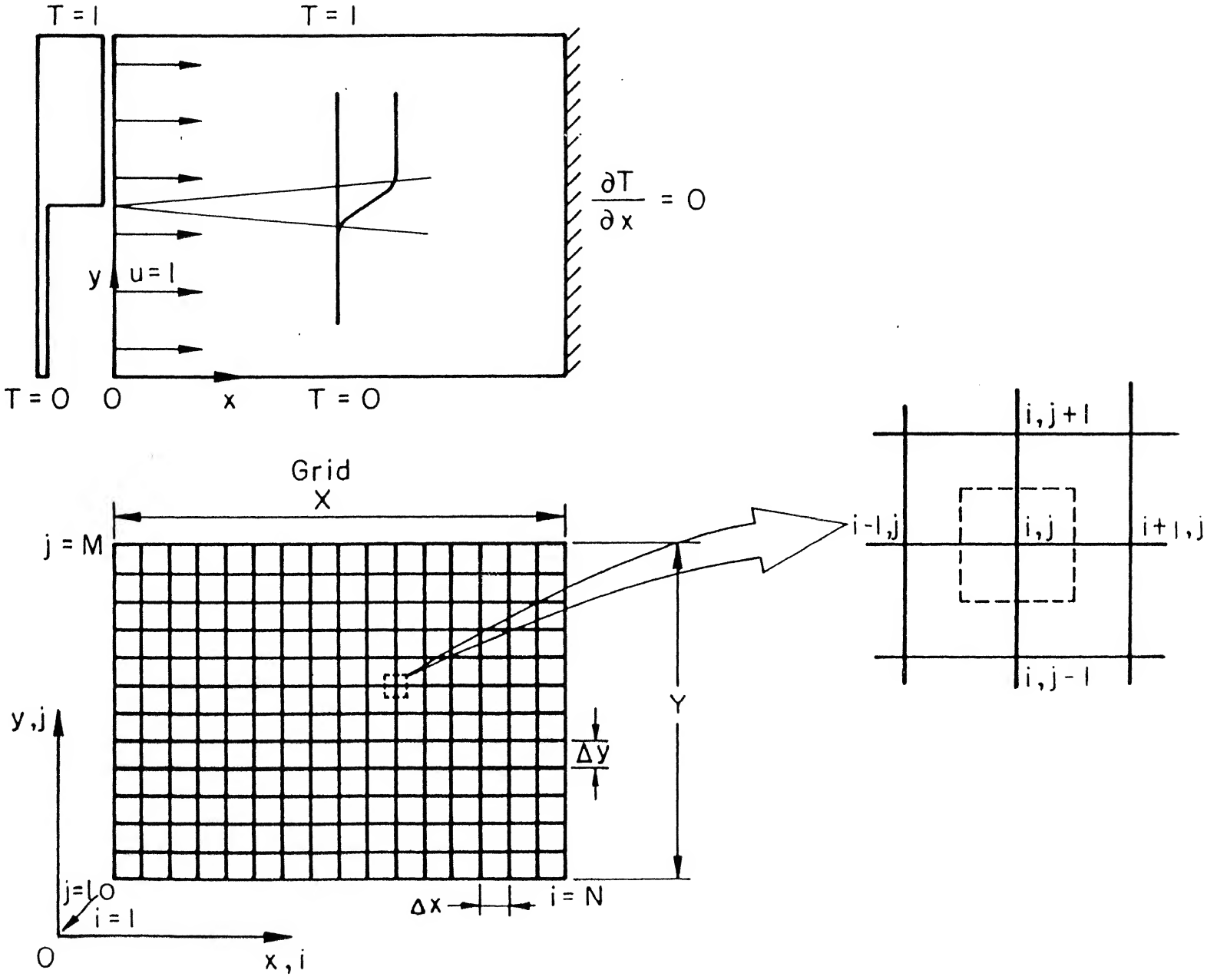


Figure 3.1 : Description of the two dimensional problem.

numbers, the mixing layer is thin and the temperature distribution is almost a discontinuity. If Pe ——> 0, the mixing layer spreads quickly to touch the boundaries.

In this problem, the flow domain taken to be square whose edge is 2 units. We have studied temperature profiles in the mixing layer at x = 0.32 and x = 0.82 at a time t = 0.5. In these calculations a 26x26 grid is used with $\Delta x = \Delta y = 0.08$ and $\Delta t = 0.1$. The grid size is chosen so as to adequately resolve the front in any given problem.

3.1.2 Upwind Formulation

The discretization of the convection term is along the same lines as that of the one dimensional problem which was discussed in section 2.2.4, Chapter 2. This is repeated here.

$$u\,T_x = \frac{u_e\,T_i - u_w\,T_{i-1}}{\Delta x}, \quad u > 0$$

$$= \frac{u_e\,T_{i+1} - u_w\,T_i}{\Delta x}, \quad u < 0$$

Since u = 1, therefore the finite difference formulation becomes,

$$\frac{T^{p+1}_{i,j} - T^{p}_{i,j}}{\Delta t} + \frac{T^{p+1}_{i,j} - T^{p+1}_{i-1,j}}{\Delta x} = \frac{1}{Pe}\left\{ \frac{T_{i+1,j}+T_{i-1,j}-2T_{i,j}}{(\Delta x)^2} + \frac{T_{i,j+1} + T_{i,j-1} - 2\,T_{i,j}}{(\Delta y)^2} \right\}^{p+1} \tag{3.2}$$

We write this equation as,

$$\left\{A\,T_{i-1,j} + B\,T_{i,j} + C\,T_{i+1,j} + D_{i,j-1} + E\,T_{i,j+1}\right\}^{p+1} = T^{p}_{i,j}$$

for each node i.

Here,

$$A = -\frac{\Delta t}{\Delta x} - \frac{\Delta t}{Pe(\Delta x)^2}, \quad B = 1 + \frac{\Delta t}{\Delta x} + \frac{2\Delta t}{Pe(\Delta x)^2} + \frac{2\Delta t}{Pe(\Delta y)^2},$$

$$C = -\frac{\Delta t}{Pe(\Delta x)^2}, \quad D = E = -\frac{\Delta t}{Pe(\Delta y)^2}$$

We have used the Gauss-Seidel method to invert the system of equations generated by Equation (3.2). Notice that the Scarborough criterion of diagonal dominance is unconditionally satisfied, i.e.,

$$|B| > |A| + |C| + |D| + |E| .$$

Hence the Gauss-Seidel iterations converge unconditionally in this problem. For any node (i,j), the iterations are generated by the formula,

$$T^{**}_{i,j} = \frac{T^{p}_{i,j} - (A\,T_{i-1,j} + C\,T_{i+1,j} + D\,T_{i,j-1} + E\,T_{i,j+1})^{*}}{B}$$

Here, the superscript '*' denotes the previous iteration value and '**', the new value.

For the right hand side boundary, where the Neumann boundary condition is imposed, we have

$$T_{i+1,j} = T_{i,j} \quad \text{at} \quad i = N.$$

This is incorporated in Eq. 3.2 to yield,

$$T^{**}_{i,j} = \frac{T^{p}_{i,j} - (A\,T_{i-1,j} + D\,T_{i,j-1} + E\,T_{i,j+1})^{*}}{B + C}$$

3.1.3 Operator Splitting Formulation (ADI)

The Equation (3.1) is split as,

Predictor: $T_t + T_x = 0$ (I) and

Corrector: $T_t = \frac{1}{Pe}(T_{xx} + T_{yy})$ (II) ,

each equation applied over the same interval Δt. The boundary conditions are relevant only over the second corrector step, as in the one dimensional problem.

Solutions of the predictor and the corrector is calculated as follows:

Step 1: The convective operator (I) has an analytical solution, T(t-x) = constant. This can be written as,

$$T^{p+1}((t + \Delta t) - x) = T^{p}(t - (x - \Delta t))$$

As in the one dimensional problem, the temperature at a point at the new time step is obtained by transferring the value prevailing at the current time at the rearward node if $\Delta t = \Delta x$. If $\Delta t \neq \Delta x$, interpolation is required.

Step 2: The corrector step consists of the two dimensional heat equation. It is solved using the efficient Alternating direction implicit (ADI) technique. The ADI method is second order accurate with a truncation error of $O\left[(\Delta t)^2, (\Delta x)^2, (\Delta y)^2\right]$.

In ADI, we first compute row by row intermediate temperatures while sweeping along the 'j' axis {See Figure 3.1}.

Then, by sweeping along the 'i' axis, final temperatures at the end of a timestep are calculated in the column by column fashion. TDMA is applicable here for computing row or column temperatures. The timestep for each sweep is $(\Delta t)_{adi} = \Delta t/2$. The discretization for each sweep is given below.

Sweep along 'j' axis:

$$T^{p+1/2}_{i,j} - T^{p}_{i,j} = \frac{(\Delta t)_{adi}}{Pe} \left\{ \left[\frac{T_{i+1,j} + T_{i-1,j} - 2T_{i,j}}{(\Delta x)^2} \right\}^{p+1/2} + \left\{ \frac{T_{i,j+1} + T_{i,j-1} - 2T_{i,j}}{(\Delta y)^2} \right\}^{p} \right]$$

or

$$\left\{ A\, T_{i-1,j} + B_{left}\, T_{i,j} + C\, T_{i+1,j} \right\}^{p+1/2} = \left\{ B_{right}\, T_{i,j} + D\, T_{i,j+1} + E\, T_{i,j-1} \right\}^{p}$$

where

$$A = C = -\frac{(\Delta t)_{adi}}{Pe\,(\Delta x)^2}, \quad B_{left} = 1 + \frac{2(\Delta t)_{adi}}{Pe(\Delta x)^2},$$

$$D = E = \frac{(\Delta t)_{adi}}{Pe(\Delta y)^2}, \quad B_{right} = 1 - \frac{2(\Delta t)_{adi}}{Pe(\Delta y)^2},$$

Sweep along 'i' axis:

$$T^{p}_{i,j} - T^{p+1/2}_{i,j} = \frac{(\Delta t)_{adi}}{Pe} \left[\left\{ \frac{T_{i+1,j} + T_{i-1,j} - 2T_{i,j}}{(\Delta x)^2} \right\}^{p+1/2} + \left\{ \frac{T_{i,j+1} + T_{i,j-1} - 2T_{i,j}}{Pe(\Delta y)^2} \right\}^{p+1} \right]$$

or,

$$\left\{A\ T_{i,j-1} + B_{left} + C\ T_{i,j+1}\right\}^{p} = \left\{B_{right}\ T_{i,j} + D\ T_{i+1,j} + E\ T_{i-1,j}\right\}^{p+1/2}$$

where

$$A = C = \frac{-(\Delta t)_{adi}}{Pe(\Delta y)^2}, \quad B_{left} = 1 + \frac{2(\Delta t)_{adi}}{Pe(\Delta y)^2},$$

$$D = E = \frac{(\Delta t)_{adi}}{Pe(\Delta x)^2}, \quad B_{right} = 1 - \frac{2(\Delta t)_{adi}}{Pe(\Delta x)^2}.$$

In the OS algorithm, successive applications of the two steps completes the calculation of temperature for a single time increment Δt.

3.1.4 The OS Algorithm (Gauss Seidel)

To check the results predicted by ADI, the diffusive corrector step has been solved independently by a Gauss-Seidel method.

The corrector step equation in finite difference form is,

$$\frac{T_{i,j}^{p+1} - T_{i,j}^{p}}{\Delta t} = \frac{1}{Pe}\left\{\frac{T_{i+1,j}+T_{i-1,j}-2T_{i,j}}{(\Delta x)^2} + \frac{T_{i,j+1}+T_{i,j-1}-2T_{i,j}}{(\Delta y)^2}\right\}^{p+1}$$

or,

$$\left\{A\ T_{i-1,j} + B\ T_{i,j} + C\ T_{i+1,j} + D\ T_{i,j-1} + E\ T_{i,j+1}\right\}^{p+1} = T_{i,j}^{p}$$

where

$$A = C = \frac{-\Delta t}{Pe(\Delta x)^2}, \quad D = E = \frac{-\Delta t}{Pe(\Delta y)^2},$$

$$B = 1 + \frac{2(\Delta t)}{Pe(\Delta x)^2} + \frac{2(\Delta t)}{Pe(\Delta y)^2}.$$

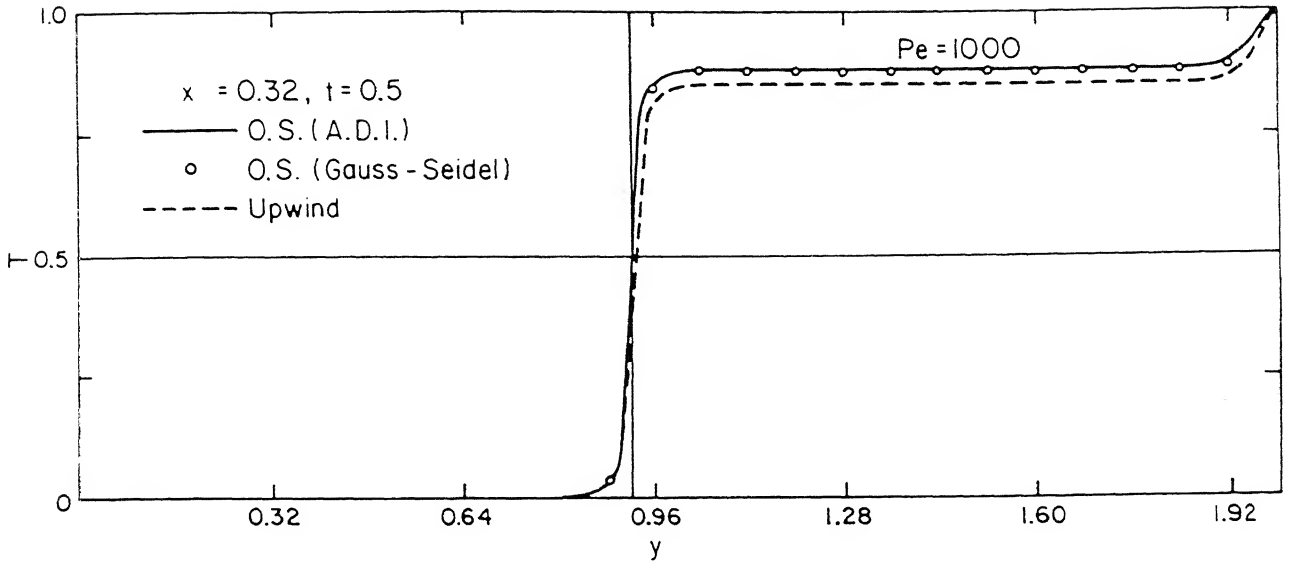
Here as in upwind formulation,

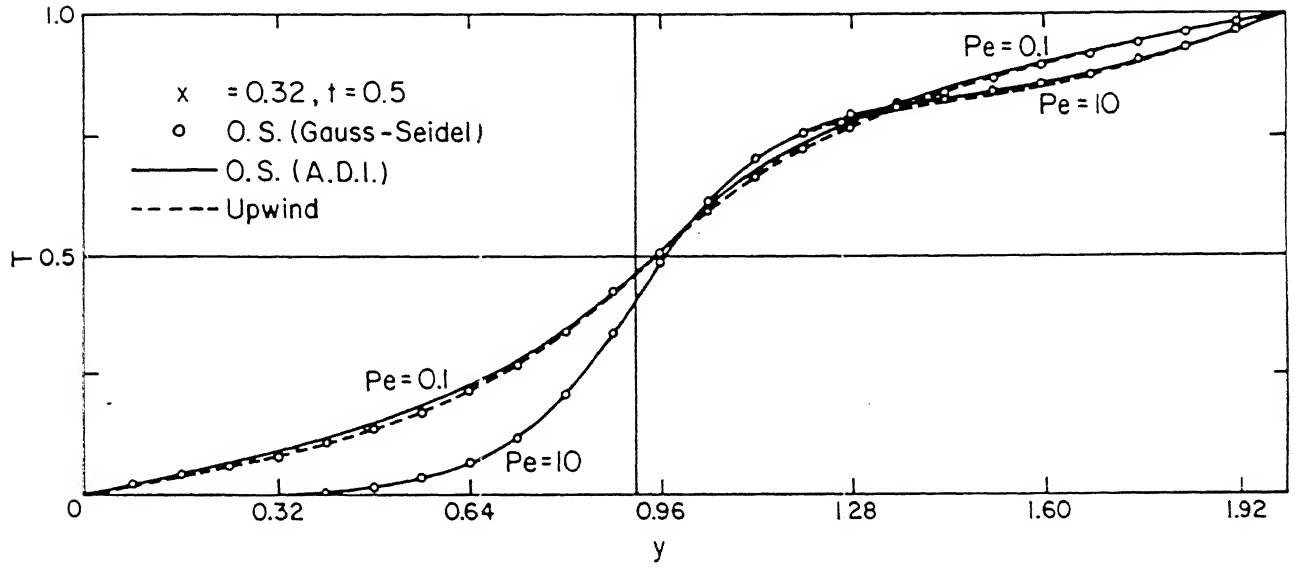
$$|B| > |A| + |C| + |D| + |E|$$

Matrix inversion proceeds along the same lines as in the upwind method (Section 3.1.2).

3.1.5 Results

We have solved the corrector step of the OS algorithm using two different numerical techniques to ascertain that the difference between the predictions by the upwind and the OS algorithms is not due to the use of different matrix inventers such as the Gauss-Seidel and the ADI methods. There is practically no difference between the three solutions namely OS (Gauss-Seidel), OS (ADI) and upwind at Pe = 0.1, 1 and 10 {See Figures 3.2 and Figure 3.3} for both cross sections (x = 0.32 and x = 0.72). At Pe = 100 and Pe = 1000, we do observe a difference between the OS and the upwind predictions. Here, the OS (ADI) profile continues to be identical to the OS (Gauss-Seidel) profile. The temperatures predicted by the upwind method are higher than those obtained from the OS algorithm. This difference can be attributed to longitudinal false diffusion arising from upwinding.





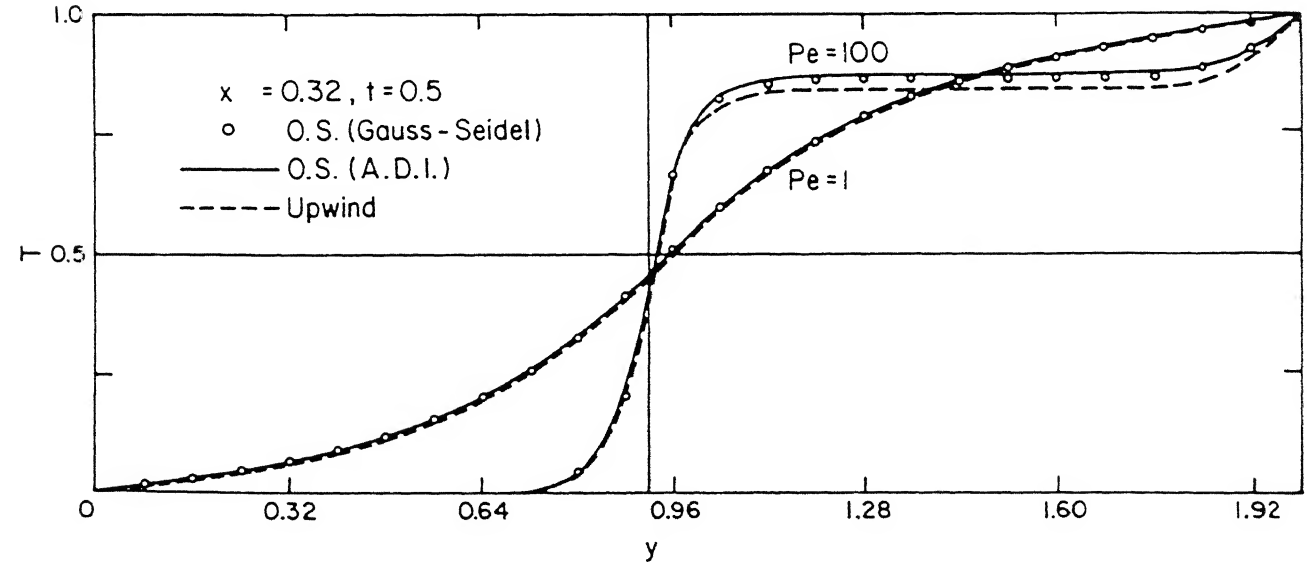


Figure 3.2 : Two dimensional problem: Pe = 0.1, 1, 10, 100, 1000 at x = 0.32.

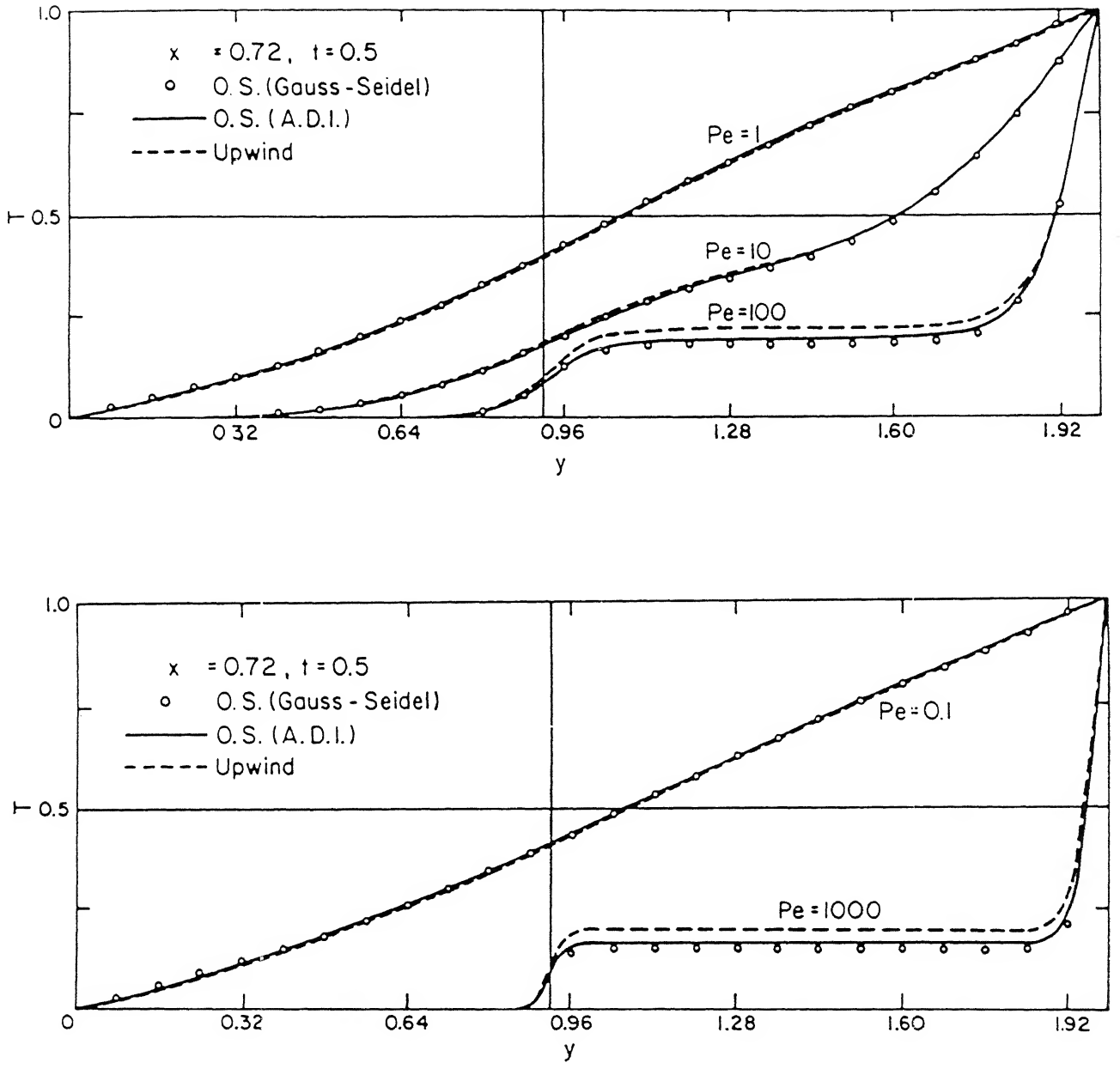


Figure 3.3 : Two dimensional problem : Pe = 0.1, 1, 10, 100, 1000 at x = 0.72.

3.2.1 Grid-Orientation Effect

In the one and two dimensional problems studied here, we observe the effect of false diffusion in high Pe flows which are aligned with the Cartesian grid. In addition to the severe truncation error in computations involving field variable curvature in the streamwise direction, there is also the problem of streamline to grid skewness in two and three dimensions [1,2].

This is called the Grid-orientation effect. Although this problem can be alleviated by skew differencing procedures [12], numerical experience indicates that the remaining truncation error is of the same order as the skewness error that has been eliminated. In addition, there is a possibility of convective instability, which makes it inadvisable to use the technique in modelling the convection of transport variables. Thus, the relative cost-benefit ratio of skew differencing is rather unfavourable.

An approximate expression for the false diffusion coefficient of the upwind scheme for a two dimensional problem has been given by deVahl-Davis and Mallinson (1972); it is

$$\Gamma_{false} = \frac{\rho\, U\, \Delta x\, \Delta y \sin 2\theta}{4\,(\Delta y \sin^3\theta + \Delta x \cos^3\theta)} \qquad [13]$$

where U is the resultant velocity, and θ is the angle between 0^0 and 90^0) made by the velocity vector with the x direction. See Appendix A for the role of Γ_{false} in causing truncation error in the upwind scheme.

Note that Γ_{false} is a maximum when $\theta = 45^{o}$. At this level of skewness, upwinding is expected to be least accurate. We demonstrate below that the OS algorithm continues to be accurate under the circumstances.

Consider the two dimensional heat transfer problem where the velocity field is not necessarily parallel to the grid. The governing equation is,

$$T_t + u\,T_x + v\,T_y = \frac{1}{Pe}\,(T_{xx} + T_{yy}) \qquad (3.3)$$

We specify

$$u \equiv \frac{1}{\sqrt{2}}\,,\quad v \equiv \frac{1}{\sqrt{2}}\,,$$

so that the flow is at 45^{o} to the grid.

The boundary conditions are as shown in Figure 3.4. The initial condition $0 < x < X$, $0 < y < Y$, $T(x,y,0) \equiv 0$ corresponds to an initially cold fluid.

This problem is identical to the two dimensional problem described in Section 3.1.1 except for the fact that the two parallel streams of equal unit velocity and of unequal temperature ($T = 0$ and $T = 1$), instead of being aligned with the grid, are inclined to it at 45^{o}. Initial temperature of the fluid in the physical domain is $T \equiv 0$ (except at the boundaries where Dirichlet boundary conditions are prescribed). Here again, a thermal mixing layer is formed. For definiteness, we study the problem at steady state. At steady state, the mixing layer thickness is inversely related to the magnitude of the Peclet number.

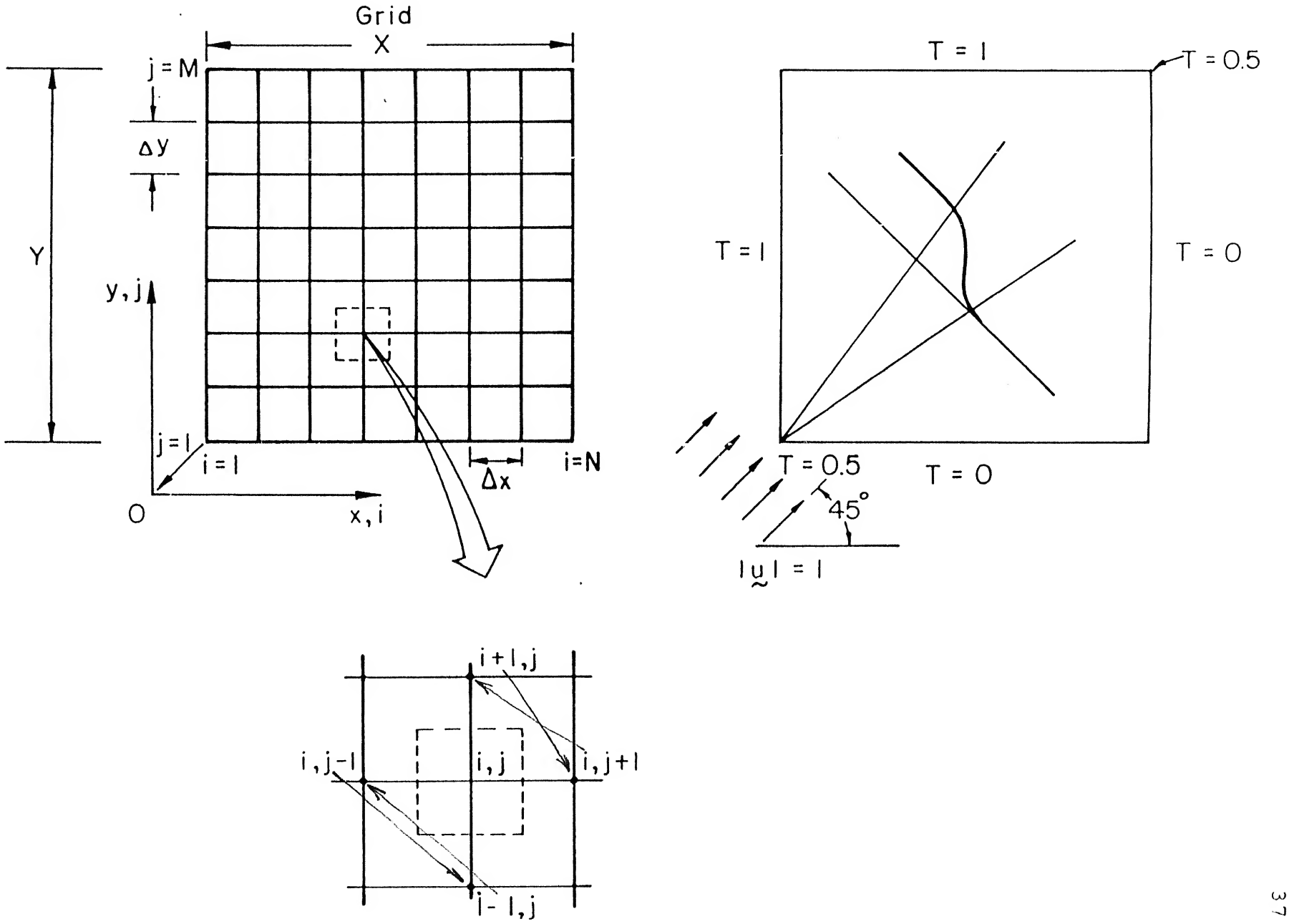


Figure 3.4 : Grid orientation effect : description of the problem.

Steady state is reached at a time level t = 2.8 units. The following data is used in the numerical computation:

$\Delta t = 0.04, \quad \Delta x = 0.04, \quad \Delta y = 0.04$

Grid size - 51x51

Region size - 2x2

3.2.2 Analytical Solution

For an infinite region, the steady state temperature profile can be obtained at high Peclet numbers by solving the reduced equation,

$$T_{\eta} = \frac{1}{Pe} T_{\xi\xi} \qquad (3.4)$$

where η and ξ are coordinates parallel and normal to the mean flow.

The boundary conditions are,

$\eta = 0, \quad \xi \geq 0, \quad T = 1$

$\eta = 0, \quad \xi < 0, \quad T = 0$

as $\xi \longrightarrow \infty, \quad T \longrightarrow 1$

as $\xi \longrightarrow -\infty, \quad T \longrightarrow 0$

The analytical solution of this problem is,

$$T(\eta,\xi) = \frac{1}{2} (1 + \text{erf}\, \Phi)$$

where

$$\Phi = \frac{\xi}{2} \sqrt{\frac{Pe}{\eta}} .$$

This is valid for Pe >> 1.

3.2.3 Upwind Formulation

Discretization of Equation 3.3, using the upwind method for the convection terms, results in the following nodal equation:

$$\frac{T_{i,j}^{p+1} - T_{i,j}^{p}}{\Delta t} + \frac{1}{\sqrt{2}} \left\{ \frac{T_{i,j} - T_{i-1,j}}{\Delta x} + \frac{T_{i,j} - T_{i,j-1}}{\Delta y} \right\}^{p+1}$$

$$= \frac{1}{Pe} \left\{ \frac{T_{i+1,j} + T_{i-1,j} - 2T_{i,j}}{(\Delta y)^2} + \frac{T_{i,j+1} + T_{i,j-1} - 2T_{i,j}}{(\Delta y)^2} \right\}^{p+1}$$

This simplifies to,

$$\left\{ A\ T_{i-1,j} + B\ T_{i,j} + C\ T_{i+1,j} + D\ T_{i,j-1} + E_{i,j+1} \right\}^{p+1} = T_{i,j}^{p}$$

for each node i, where

$$A = -\frac{\Delta t}{\sqrt{2}(\Delta x)} - \frac{\Delta t}{Pe(\Delta x)^2}, \quad B = 1 + \frac{\Delta t}{\sqrt{2}(\Delta x)} + \frac{\Delta t}{\sqrt{2}(\Delta y)} + \frac{2(\Delta t)}{Pe(\Delta x)^2} + \frac{2(\Delta t)}{Pe(\Delta y)^2}$$

$$C = -\frac{\Delta t}{Pe(\Delta x)^2}, \quad D = \frac{-\Delta t}{\sqrt{2}\ (\Delta y)} - \frac{\Delta t}{Pe(\Delta y)^2}, \quad E = -\frac{\Delta t}{Pe(\Delta y)^2}$$

The Gauss-Seidel method is used to solve for $T_{i,j}^{p}$ at all the internal nodes.

3.2.4 Operator Splitting Formulation (ADI)

The governing equation (3.3) is split as,

(I) $T_t + \frac{1}{\sqrt{2}} T_x + \frac{1}{\sqrt{2}} T_y = 0$ (predictor) and

(II) $T_t = \frac{1}{Pe} (T_{xx} + T_{yy})$ (corrector).

The analytical solution of step I is obtained by using a new coordinate $\phi = \frac{1}{\sqrt{2}} (x + y)$.

This result in,

$T_t + T_\phi = 0$ whose solution is $T(t - \phi) =$ constant.

This can be written as

$$T^{p+1} (t + \Delta t - \phi) = T^p (t - (\phi - \Delta t)).$$

For $\Delta x = \Delta y$ and $\Delta t = \Delta\phi$, this means that the temperature at a point at the new timestep is obtained by transferring the value prevailing at the current time at the diagonally opposite upstream node, i.e.,

$$T^{p+1}_{i,j} = T^p_{i-1,j-1}$$

The corrector step II is solved as in Section 3.1.3 using A.D.I.

3.2.5 Operator Splitting Formulation (Gauss-Seidel)

Here step II is solved by the Gauss-Seidel technique instead of ADI. See Section 3.1.4 for details.

3.2.6 Results

Comparison of profiles obtained by the analytical solution, upwinding, OS (ADI) and OS(Gauss-Seidel) is shown in Figure 3.5. The analytical solution cannot be used for comparison at low Peclet numbers such as Pe = 10 and Pe = 100 due to

a) the inapplicability of the boundary-layer approximation and b) the finite size of the region considered in numerical

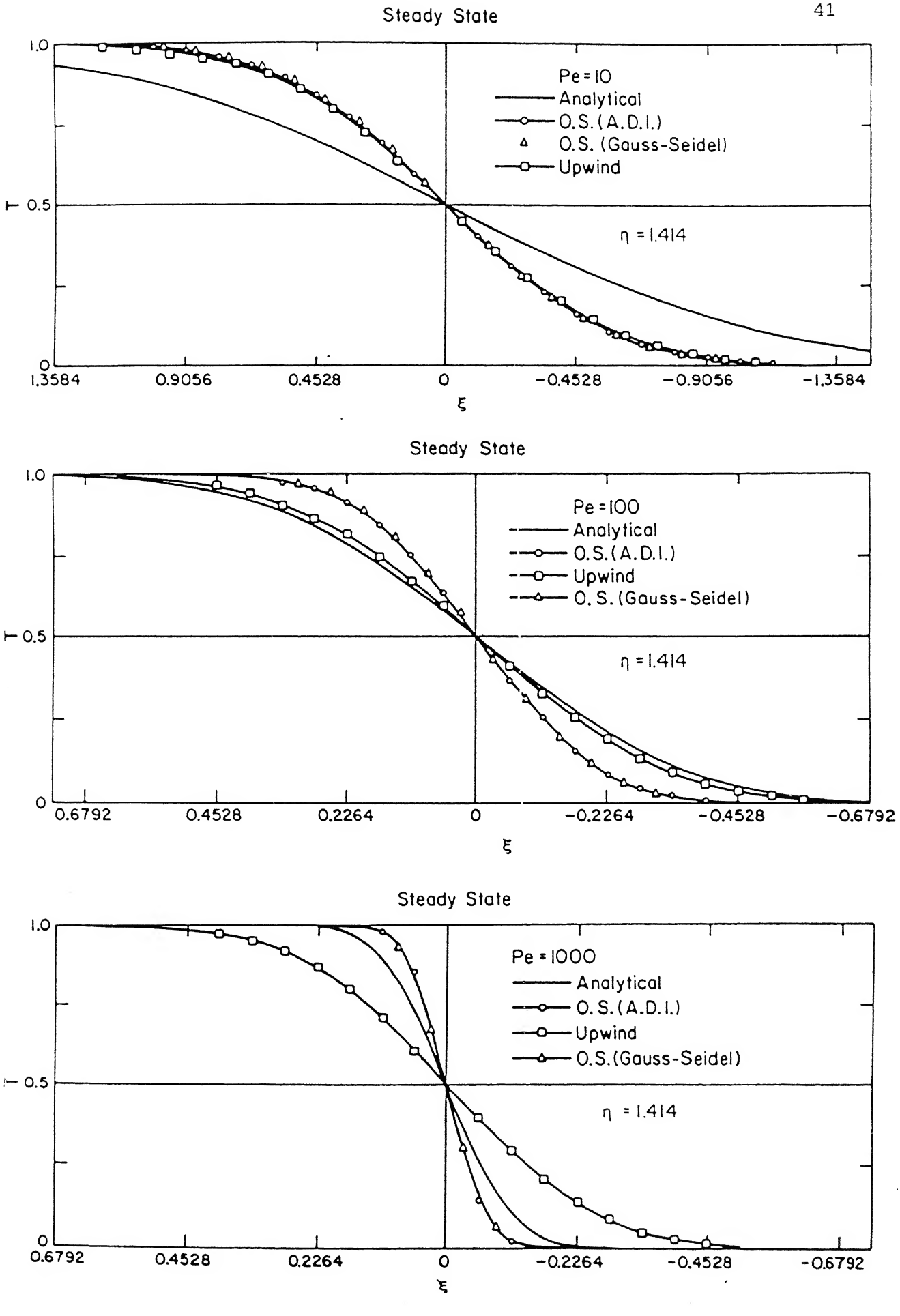


Figure 3.5 : Grid orientation effect : Pe = 10, 100, 1000.

simulation. At Pe = 1000, the mixing layer is thin and temperature reaches values 1 and 0 well within the flow domain.

At Pe = 10, both upwinding and operator splitting give identical results. The temperature profile becomes sharper at Pe = 100 and this is well reproduced by the OS solution. For Pe = 1000, we find that OS results come close to the analytical result. Note that at Pe = 100 and Pe = 1000, the upwind results are almost identical. We conclude that for the grid used, false diffusion dominates over physical diffusion in the upwind solutions at Pe > 100.

We also notice that OS (ADI) and OS (Gauss-Seidel) profiles are almost identical in the three cases. Hence we are certain that the differences in the O.S. and the upwind results are due to false diffusion rather than differences in the Gauss-Seidel and ADI methods.

For the fully evolved mixing layer, it can be shown that the mixing layer thickness varies as

$$\delta \sqrt{Pe} = \text{constant}.$$

In Table 1, the $\delta \sqrt{Pe}$ values as predicted by the two schemes are given for three Peclet numbers. We see that upwind scheme values come nowhere close to being a constant. OS scheme shows much better response vis-a-vis the analytical solution.

In Table 2, $\delta \sqrt{Pe}$ values predicted by OS method for different grid sizes is listed. We observe that despite a change in the magnitude of the constant, $\delta\sqrt{Pe}$ remains nearly constant. This means that an increase in Pe reduces δ in the right proportion and produces a suitably sharp temperature profile.

Table 1 : Comparison of $\delta\sqrt{Pe}$ Values

Pe	Analytical	Upwind	OS
10	13.01	6.86	6.66
100	13.08	19.32	7.81
1000	12.88	30.173	8.95

Table 2 : Effect of Grid-coarsening on $\delta\sqrt{Pe}$ Values

Pe	N(=M) = 51 Δx(=Δy) = 0.04	N(=M) = 41 Δx(=Δy) = 0.05	N(=M) = 31 Δx(=Δy) = 0.067
10	6.66	6.64	6.614
100	7.81	7.954	8.014
1000	8.95	7.826	10.14

CHAPTER 4

NUMERICAL MODELING OF HOT-WATER INJECTION METHOD IN OIL RECOVERY

4.1 Introduction

Before we start with the application of operator splitting algorithm in the numerical simulation of the hot-water injection method, a brief description of various oil recovery processes is first presented [16]. A common misconception about hydrocarbon recovery is that oil or gas is found in large pools in underground caverns and must be pumped out of the reservoir in a manner similar to pumping a liquid out of a storage tank. This is not the case. In general, the hydrocarbon is trapped in the microscopic pores of a rock, like sandstone, and will flow through the rock only under the influence of extremely large pressure differentials. A large percentage of pores are connected and the fluids can flow through these linked pore paths. However, the paths are very small, irregular and twisted.

In many instances, the pressure found in reservoirs are extremely high. Often these pressures are high enough to force the resident fluids to flow through the porous medium and out of the wells without any pumping effort at the wells. As the reservoir pressure is depleted through fluid extraction, the hydrocarbon production decreases and finally stops. This type of recovery, termed primary recovery, extracts 15-30% of hydrocarbons in the reservor.

Subsequently, a fluid such as water may be injected into some wells in the reservoir while petroleum is produced through others. This serves the dual purpose of maintaining high reservoir pressures and flow rates and also of flooding the porous medium to physically displace some of the oil and push it towards the production wells. This type of pressure maintenance and waterflooding is usually termed secondary recovery.

Unfortunately, waterflooding is still not extremely effective and significant amounts, upto 50%, of hydrocarbon often remain in the reservior. Due to strong surface tension effects, large amounts of oil are trapped in small pores with narrow throats and cannot be washed out with routine waterflooding techniques.

In order to recover more of the residual hydrocarbons, several enhanced recovery techniques involving complex chemical and thermal effects have been developed. These techniques form part of a variety of methods termed tertiary recovery and have been given the name enhanced oil re c overy (EOR) techniques. Since oil is trapped in the pores of the reservoir because of surface tension, several EOR techniques involve the lowering of the surface tension to allow the oil to flow from the small pores. Some of these techniques are given below.

1) Surfactants flooding: This process involves the use of a surface active chemical which gets into the pores and causes reduction in the interfacial tension between oil and water. This way capillary forces are reduced and it leads to improved oil production.

2) Miscible flooding: It involve injection of gases such as CO_2

and other chemical solavents which mix with the resident hydrocarbon. Together they form one fluid phase of lower surface tension. Once the mixing or "miscibility" is attained, the fluids will flow together in one phase, eliminating distinction between phases, and enhanced hydrocarbon recovery is possible.

3) in situ combustion: When the hydrocarbon reserves are too deep to be mined, techniques for in situ transformation of the hydrocarbons into states which can be pumped to the surface via production wells are required. In this technique, the solid hydrocarbon is ignited in the ground and oxygen or air is pumped into the injection wells to maintain combustion. The hot hydrocarbon gases produced in the combustion process are pumped to the surface via the production wells for use as low-grade hydrocarbon.

4) Thermal recovery methods: This techniques for lowering the viscosity of heavy oils involves addition of very high temperatures to effectively "melt" the viscous hydrocarbons, allowing them to flow more readily through the porous medium. A primary technique for thermal flooding is to inject either hot water or steam into the reservoir via certain injection wells. As the heated fluids come into contact with the cooler oil, the temperature is raised. Consequently the oil viscosity is reduced and it can be pushed easily through the porous media. Getting the high temperature fluid down into the reservoir is a difficult engineering problem. The hot injecting fluid may lose heat while moving through the injection well and later to the host rock bounding the oil

rich region. Some energy is also consumed in the chemical reactions of hydrocarbons. Apart from heat loss, there is also the problem of gravity override. This occurs when the hot fluid rises on to the top of the oil saturated sand layer and bypasses it. If the flow rate is high, the interface between the resident petroleum and the invading water can become unstable leading to the formation of long fingers. These fingers grow in length towards the production wells, bypassing much of the oil. These factors cause a drastic reduction in oil production.

It is this last type of EOR method based on thermal recovery that we are concerned with in this chapter. We numerically study EOR using hot water in a small two dimensional rectangular test cell. The operator splitting technique is used to solve the energy equation.

4.2 Literature Review

Hot water injection technique is a tested technique used in petroleum industry. Many attempts have been made in the past to model the fluid and heat flow involved in this problem. Researchers have taken recourse to analytical methods to predict the movement of saturation and temperature fronts in the oil fields. Lauwerier's [21] attempt to formulate the equation of heat balance and determine its analytical solution for a simple real life situation is one of the earliest efforts in this direction. Spillette [20] has refined Lauwerier's approach and extended it to steam injection as well. There are a variety of theoretical methods listed in References [16] and [17], but they are essentially extensions of Lauwerier's and Spillette's works.

Despite the elegance and apparent simplicity of analytical solutions, they are plagued with problems. Analytical solutions are possible only for very simple physical situations. When real life difficulties such as heterogeneties in the soil, multiphase multicomponent fluid flow, distillation effect of hydrocarbons are taken into account, no analytical solutions are possible. At best, the analytical approach can yield initial estimates of the parameters in the physical processes. Numerical modelling of the reservoir process is an answer to the limitations of the analytical approach. The computational approach allows us to include as much physics as possible of the processes involved. A large number of coupled nonlinear partial differential equations can be solved simultaneously by numerical methods. Heterogeneties and geometrical complexities present in any reservoir can be taken into account. Moreover it produces detailed information about the process which is useful to the oil engineers. As a result of these advantages, numerical modeling has rapidly gained importance.

Reference [19] gives an excellent introduction to the general topic of reservoir simulation. Description of various oil recovery methods is given by Ewing [16]. It is also a good source of references of important research papers in this area. Boberg [17] describes thermal methods of oil recovery in detail and presents an extensive compilation of case studies and field data. He also presents two state-of-art numerical simulators [22] and [23] for steam injection process. An important paper by Stevenson et al [24] describes some of the latest computational methods in reservoir simulation.

4.3 Nomenclature

x,y	Global grid coordinates m
Δx Δy	Grid size along x and y axes respectively m
X,Y	Overall domain dimensions along x and y axes respectively m
X_d,Y_d	Problem domain dimensions along x and y axes respectively m
i,j	Node indices along x and y axes respectively
$\hat{i}$, $\hat{j}$	Unit vectors along x and y axes respectively
t	Time s
Δt	Step of time s
p	Current timestep
ϵ	Porosity
$S_{o,w}$	Saturation: oil, water
$\xi_{o,w}$	Compressibility: oil, water $1/p_a$
$\beta_{o,w}$	Expansivity: oil, water $1/^oC$
$\rho_{o,w}$	Density: oil, water kg/m^3
k	Absolute permeability of porous medium darcies
$k_{ro,w}$	Relative permeability: oil, water
K_h	Thermal conductivity of the porous medium $W/m-^oC$
$p_{o,w}$	Phase pressure : oil, water Pa
$\mu_{o,w}$	Viscosity: oil, water Pa-s
$p_{c_{ow}}$	Capillary pressure of oil water system Pa
$c_{o,w}$	Specific heat capacity: oil, water $J/kg——^oC$
$(\rho c)_w$	Heat capacity/volume of the porous medium J/m^3-^oC
$\tilde{v}_{o,w}$	Darcy velocity: oil, water m/s
v_{ox}	Darcy velocity of oil: x component m/s

v_{oy}	Darcy velocity of oil: y component m/s
v_{wx}	Darcy velocity of water: x component m/s
v_{wy}	Darcy velocity of water: y-compoanent m/s
T	Temperature ^{o}C
T_f	Formation temperature ^{o}C
T_g	Generation temperature at source ^{o}C
U	Composite velocity: x direction m/s
V	Composite velocity: y direction m/s

4.4 Definitions

Here we explain some of the terms used in this chapter.

(a) Porosity: It is the ratio of total pore volume to the total volume of porous medium (sand).

(b) Percentage pore volume (ppv): It is an index of efficiency of oil recovery. ppv is the ratio of total volume of oil produced to the total pore volume. It is calculated as

$$ppv = \frac{\text{Initial volume of oil} - \text{Present volume of oil}}{\text{Pore volume}} \times 100$$

$$= \frac{\sum_{i=1}^{nd} \sum_{j=1}^{M} \varepsilon \left[S_o(i,j)\big|_{initial} - S_o(i,j)\big|_{present} \right]}{\sum_{i=1}^{nd} \sum_{j=1}^{M} \varepsilon} \times 100$$

$$= \frac{\sum_{i=1}^{nd} \sum_{j=1}^{M} \left(S_o(i,j)\big|_{initial} - S_o(i,j)\big|_{present} \right)}{(nd \cdot M)} \times 100$$

where S_o is oil saturation.

(c) Oil and water velocities: $\tilde{v}_o$ and $\tilde{v}_w$ are the volume averaged velocities called Darcy velocities and are related to the local-fluid velocities as follows:

$$\tilde{v}_o = S_o \varepsilon \tilde{V}_o, \quad \tilde{v}_w = S_w \varepsilon \cdot \tilde{V}_w$$

(d) Saturation: It is the fraction of the space available to flow (i.e. pore volume) occupied by a phase.

(e) Biot number (Bi): It is an index of heat loss to the host rock bounding the oil rich sand { See Figure 4.1 }. Higher the value of Bi, higher the heat loss.

(f) Oil displacement efficiency: is the efficiency with which oil is flushed out from the porous medium by water. It is directly related to ppv. Higher the ppv, higher the output of oil and consequently higher the oil displacement efficiency.

(g) Overhang: It is the impermeable rock lying on the top of the porous medium {See Figure 4.1} and is also called "overburden".

(h) Underhang: impermeable rock lying underneath the porous medium and is also called "under burden".

(i) Formation: refers to the whole structure consisting of host rock and oil rich sand.

4.5 Physical Description

The physical domain of the two dimensional problem is shown in the Figure 4.1. Essentially, we would be studying the water piston effect in a rectangular domain packed with sand and guarded by impermeable but thermally conductive rocks above and below the porous region (called overhang and underhang rerpectively). Initially the porous medium (i.e. sand) is impregnated with oil to a given saturation. It has some initial temperature T_f (called as formation temperature) and is at rest. We assume in this study that the pore volume not containing oil has water. At equilibrium, the oil and water pressures in sand

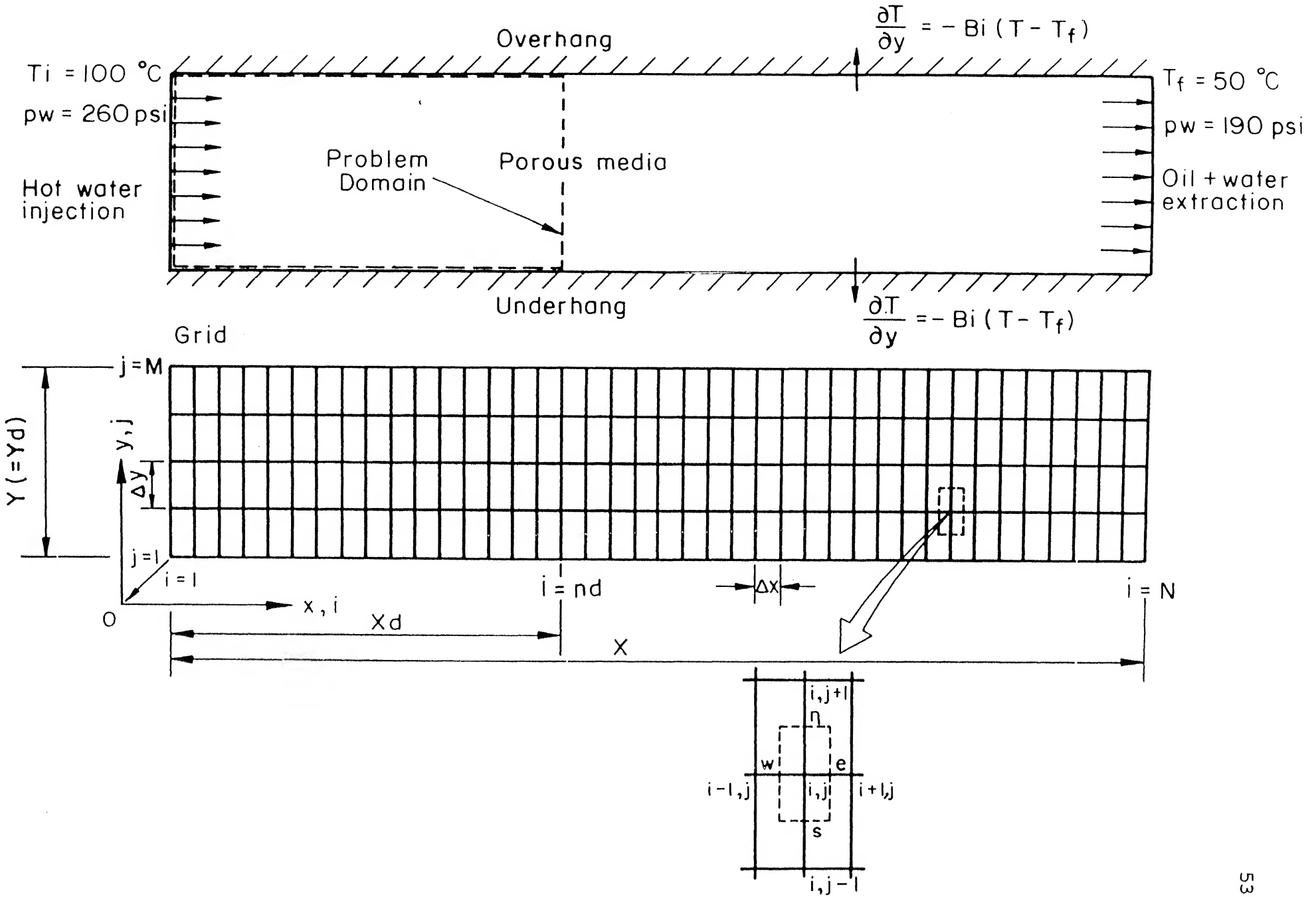


Figure 4.1 : Description of the oil recovery problem.

are uniquely determined from their saturation. Oil recovery is initiated by pushing water at a given higher pressure and a given temperature T_i. We observe the movement of saturation and temperature profiles in the porous region. These profiles would have the shape of fronts because the Peclet number in this problem is quite high {See Appendix B(b)}. The following two variants of the main physical phenomena are addressed to in this study.

(i) Isothermal Problem: Here the water that is being pushed in is of the same temperature as that of the formation (T_f). Therefore, we need not solve the energy equation.

(ii) Non-isothermal Problem: Heat is transported into the flow domain through hot water that is pushed in. Here, the temperature distribution is important and the energy equation must be solved along with the pressure equations.

4.6 Scope of this Chapter

Results have been presented in this chapter for the following cases.

(i) Effects of formation temperature (T_f) on oil recovery (ppv).

(ii) Comparison of oil recovery for the isothermal and non-isothermal cases for a given value of T_f.

(iii) Evolution of saturation, temperature and velocity profiles for the isothermal and non-isothermal cases.

(iv) Effect of heat loss, to the overhang and the underhang, on oil recovery.

4.7 Governing Differential Equations

Since the phenomena of two phase flow through the porous media is too complicated to be described on a microscopic scale equations applicable on a macroscopic scale have been used in the present study {See Appendix D for a discussion on the macroscopic approach}. These equations are derived in Appendix E.The pressure equations for oil and water phases are,

$$\text{Oil:}\quad -S_o\beta_o\frac{\partial T}{\partial t}+S_o\xi_o\frac{\partial p_o}{\partial t}-\frac{dS_w}{dp_{c_{ow}}}\left(\frac{\partial p_o}{\partial t}-\frac{\partial p_w}{\partial t}\right)$$

$$=\frac{1}{\rho_o}\nabla\cdot\left(-\frac{k\,k_{ro}\,\rho_o}{\mu_o\varepsilon}\right)\nabla p_o \qquad (4.1)$$

$$\text{Water:}\quad -S_w\beta_w\frac{\partial T}{\partial t}+S_w\xi_w\frac{\partial p_w}{\partial t}+\frac{dS_w}{dp_{c_{ow}}}\left(\frac{\partial p_o}{\partial t}-\frac{\partial p_w}{\partial t}\right)$$

$$=\frac{1}{\rho_w}\nabla\cdot\left(\frac{k\,k_{rw}\,\rho_w}{\mu_w\varepsilon}\right)\nabla p_w \qquad (4.2)$$

Equations 4.1and 4.2 are strongly nonlinear coupled partial differential equations. Their form is that of a parabolic heat conduction equation with source terms

$$-S_w\beta_w\frac{\partial T}{\partial t}\quad\text{and}\quad\frac{dS_w}{dp_{c_{ow}}}\left(\frac{\partial p_o}{\partial t}-\frac{\partial p_w}{\partial t}\right).$$

Temperature is determined from the energy equation,

$$\frac{\partial T}{\partial t} + U \frac{\partial T}{\partial x} + V \frac{\partial T}{\partial y} = \left(\frac{K_h}{\sigma_T}\right) \nabla^2 T \qquad (4.3)$$

Here

$$U = \frac{v_{ox}\, \rho_o\, c_o + v_{wx}\, \rho_w\, c_w}{\varepsilon \cdot \left\{(1-S_w)\, \rho_o c_o + S_w\, \rho_w\, c_w\right\} + (1-\varepsilon)(\rho c)_R}$$

$$V = \frac{v_{oy}\, \rho_o\, c_o + v_{wy}\, \rho_w\, c_w}{\varepsilon \cdot \left\{(1-S_w)\, \rho_o c_o + S_w\, \rho_w c_w\right\} + (1-\varepsilon)(\rho c)_R}$$

The following equations are also valid.

$$p_{c_{ow}} = (S_w) = p_o - p_w \qquad (4.4)$$

$$S_o + S_w = 1 \qquad (4.5)$$

Velocities are determined using Darcy's law as,

$$\tilde{v}_o = -\frac{k\, k_{ro}}{\mu_o} (\nabla p_o)$$

$$\tilde{v}_w = -\frac{k\, k_{rw}}{\mu_w} (\nabla p_w) \qquad (4.6)$$

The equations of state are,

$$\rho_o = \rho_{o,ref} \left\{ 1 + \xi_o\, (p_o - p_{o,ref}) - \beta_o (T - T_{ref}) \right\} \qquad (4.7)$$

$$\rho_w = \rho_{w,ref} \left\{ 1 + \xi_w (p_w - p_{w,ref}) - \beta_w (T - T_{ref}) \right\}$$

$$\mu_{o,w} = \mu_{o,w} (T)$$

$$k_{ro,w} = k_{ro,w} (S_w)$$

$$p_{c_{ow}} = p_{c_{ow}} (S_w)$$

See Table 4 for the dependence of $\mu_{o,w}$, $k_{ro,w}$, $p_{c_{ow}}$ on T, S_w, S_w respectively.

4.8 Initial and Boundary Conditions

The initial and boundary conditions, along with much of thedata { Table 3} is adapted from Boberg [17], p.24. These values have been used by Boberg for a laboratory test where the dimensions of the test cell were of the same order as the domain dimensions of this problem.

At the left boundary of the domain {See Figure 4.1}, water at a pressure of 190 psi and at a temperature of 100°C is applied. Saturation of water is 0.86 here. {It is not unity because of some residual oil which can not be flushed out}. It is the maximum water saturation possible anywhere in the domain. At the right hand boundary of the porous region, water pressure, temperature and saturation are maintained at 190 psi, 50°C and 0.2 respectively.

Initially, the porous region is maintained at a uniform water pressure and temperature of 190 psi and 50°C respectively.

Oil saturation is 0.8. Oil pressure is obtained by adding water pressure to the capillary pressure, the latter being a unique function of water saturation.

The initial conditions of p_w, p_o, S_w and T were smoothened out to avoid the computational instabilities resulting from a step like initial condition. These conditions are given below.

At time $t = 0_-$,

$$p_w(x,y) = (260 - 175\,x) \text{ psi}; \quad 0 \le x \le 0.1\,X$$
$$= 190 \text{ psi} \quad ; \quad 0.1X < x \le X$$
$$S_w(x,y) = 0.86 - 1.65x \quad ; \quad 0 \le x \le 0.1\,X$$
$$= 0.2 \quad ; \quad 0.1\,X < x \le X$$
$$T(x,y) = (100-125x)^{o}C \quad ; \quad 0 \le x \le 0.1\,X$$
$$= 50\ ^{o}C \quad ; \quad 0.1\,X < x \le X$$

At time $t > 0$, the impermeability conditions at the bounding surfaces of the host rock give,

$$v_{oy} = v_{wy} = 0, \quad \text{at} \quad y = 0, Y$$

Using Darcy's law {Equation 4.6, Section 4.7}, the above impermeability condition is stated in terms of pressure as,

$$\frac{\partial p_o}{\partial y} = \frac{\partial p_w}{\partial y} = 0 \quad \text{at} \quad y = 0, \quad Y.$$

Further, at $x = 0$,

$$p_w(0,y) = 260 \text{ psi} \quad (= 1.793 \text{ MPa})$$
$$S_w(0,y) = 0.86$$
$$T(0,y) = 100^{o}C$$
$$p_o(0,y) = p_w(0,y) + p_{c_{ow}}(S_w(0,y))$$

At x = X,

$$p_w(X, y) = 190 \text{ psi} \quad (= 1.31 \text{ MPa})$$

$$S_w(X, y) = 0.2$$

$$T(X,y) = 50^{o}C$$

$$p_o(X,y) = p_w(X,y) + p_{c_{ow}}(S_w(X,y))$$

Boundary conditions related to heat loss to overhang and underhang is given by:

$$\frac{\partial t}{\partial y} + Bi.(T - T_f) = 0 \quad \text{at } y = 0, Y.$$

4.9 Discretization of Pressure Equations

The oil pressure equation {Section 4.7, Equation 4.1} can be written as:

$$-S_o\beta_o \frac{\partial T}{\partial t} + (S_o \xi_o - \frac{dS_w}{dp_{c_{ow}}}) \frac{\partial p_o}{\partial t} + (\frac{dS_w}{dp_{c_{ow}}}) \frac{\partial p_w}{\partial t}$$

$$= \frac{1}{\rho_o} \nabla . (\frac{k\, k_{ro}\, \rho_o}{\varepsilon \mu_o}) \nabla p_o$$

The discretization of equation is fully implicit in time, i.e., all spatial derivative are evaluated at the new time level.

$$\text{Let } \theta = \frac{k\, k_{ro}\, \rho_o}{\varepsilon\, \mu_o} = \frac{k \cdot k_{ro}(S_w) \cdot \rho_o(p_o, T)}{\varepsilon . \mu_o(T)} ,$$

$$\text{Then } \frac{1}{\rho_o}\,\tilde{\nabla}\cdot\theta\,\tilde{\nabla}\,p_o = \frac{1}{\rho_{o_{i,j}}}\,\frac{\left[\theta\,\frac{\partial p_o}{\partial x}\right]_e - \left[\theta\,\frac{\partial p_o}{\partial x}\right]_w}{\Delta x} + \frac{1}{\rho_{o_{ij}}}\,\frac{\left[\theta\,\frac{\partial p_o}{\partial y}\right]_n - \left[\theta\,\frac{\partial p_o}{\partial y}\right]_s}{\Delta y}$$

$$= \frac{1}{\rho_{o_{i,j}}}\cdot\frac{\theta_e\left\{\frac{p_{o_{i+1,j}} - p_{o_{i,j}}}{\Delta x}\right\}^{p+1} - \theta_w\left\{\frac{p_{o_{i,j}} - p_{o_{i-1,j}}}{\Delta x}\right\}^{p+1}}{\Delta x} + \frac{1}{\rho_{o_{i,j}}}\,\frac{\theta_n\left\{\frac{p_{o_{i,j+1}} - p_{o_{i,j}}}{\Delta y}\right\}^{p+1} - \theta_s\left\{\frac{p_{o_{i,j}} - p_{o_{i,j-1}}}{\Delta y}\right\}^{p+1}}{\Delta y}$$

Here θ_e, θ_w, θ_n, θ_s are evaluated as harmonic averages of their values at nodes that lie on their side (Figure 4.1).

The left side of the pressure equation is discretized as,

$$\left(S_o\,\xi_o - \frac{d\,S_w}{dp_{c_{ow}}}\right)_{i,j}\cdot\left[\frac{p_{o_{i,j}}^{p+1} - p_{o_{i,j}}^{p}}{\Delta t}\right] + \left(\frac{dS_w}{dp_{c_{ow}}}\right)_{i,j}\cdot\left[\frac{p_{w_{i,j}}^{p+1} - p_{w_{i,j}}^{p}}{\Delta t}\right] - S_o\beta_o\left[\frac{T_{i,j}^{p} - T_{i,j}^{p-1}}{\Delta t}\right]$$

The full equation in discretized form is written as,

$$\left[A_o p_{o_{i+1,j}} + B_o\,p_{o_{i,j}} + C_o\,p_{w_{i,j}} + D_o\,p_{o_{i-1,j}} + E_o\,p_{o_{i,j+1}}\right.$$

$$+ F_o \, p_{o_{i,j-1}} \Big]^{p+1} = G_o \qquad (4.8)$$

where

$$A_o = - \left\{ \frac{\theta_e}{\rho_{o_{i,j}} (\Delta x)^2} \right\}^p , \quad C_o = \left[\frac{dS_w}{dp_{c_{ow}}} \right]^p_{i,j} \cdot \frac{1}{\Delta t} ,$$

$$B_o = \frac{S^p_{o_{i,j}} \xi_o - \left(\frac{dS_w}{dp_{c_{ow}}}\right)^p_{i,j}}{\Delta t} + \frac{1}{\rho^p_{o_{i,j}}} \cdot$$

$$\left\{ \frac{\theta_e + \theta_w}{(\Delta x)^2} + \frac{\theta_n + \theta_s}{(\Delta y)^2} \right\}^p ,$$

$$D_o = - \left\{ \frac{\theta_w}{\rho_{o_{i,j}} (\Delta x)^2} \right\}^p , \quad E_o = - \left\{ \frac{\theta_n}{\rho_{o_{i,j}} (\Delta y)^2} \right\}^p ,$$

$$F_o = - \left\{ \frac{\theta_s}{\rho_{o_{i,j}} (\Delta y)^2} \right\}^p , \quad G_o = \left\{ \frac{S_{o_{i,j}} \xi_o - \frac{dS_w}{dp_{c_{ow}}}\Big|_{i,j}}{\Delta t} \cdot p_{o_{i,j}} \right\}^p .$$

$$+ \left\{ \frac{1}{\Delta t} \left[\frac{dS_w}{dp_{c_{ow}}} \right]_{i,j} p_{w_{i,j}} \right\}^p + S^p_{o_{i,j}} \cdot \beta \frac{T^p_{i,j} - T^{p-1}_{i,j}}{\Delta t}$$

Backward differencing in time of the temperature term is justified because the main effect of T on pressure distribution is through fluid properties. These do not change as rapidly with time as the primary variables $p_{o,w}$ and S.

The water pressure equation {Section 4.7, Equation 4.2} is also discretized as:

$$\left[A_w\, p_{w_{i+1,j}} + B_w\, p_{w_{i,j}} + C_w\, p_{o_{i,j}} + D_w\, p_{w_{i-1,j}} + E_w p_{w_{i,j+1}} + F_{w_{i,j-1}} \right]^{p+1} = G_w \qquad (4.9)$$

where

$$A_w = -\left\{ \frac{\gamma_e}{\rho_{w_{i,j}} (\Delta x)^2} \right\}^p , \quad C_w = \left\{ \frac{dS_w}{dp_{c_{ow}}} \cdot \frac{1}{\Delta t} \right\}^p ,$$

$$D_w = -\left\{ \frac{\gamma_w}{\rho_{w_{i,j}} (\Delta x)^2} \right\}^p , \quad B_w = \left\{ \frac{S_{w_{i,j}} \cdot \xi_w - \left(\frac{dS_w}{dp_{c_{ow}}} \right)_{i,j}}{\Delta t} \right\}^p$$

$$+ \frac{1}{\rho_{w_{i,j}}} \left\{ \frac{\gamma_e + \gamma_w}{(\Delta x)^2} + \frac{\gamma_n + \gamma_s}{(\Delta y)^2} \right\}^p ,$$

$$E_w = -\left\{ \frac{\gamma_s}{\rho_{w_{i,j}} (\Delta y)^2} \right\}^p , \quad F_w = -\left\{ \frac{\gamma_s}{\rho_{w_{i,j}} (\Delta y)^2} \right\}^p ,$$

$$G_w = \left\{ \frac{S_{w_{i,j}} \xi_w - \left(\frac{dS_w}{dP_{c_{ow}}} \right)_{i,j}}{\Delta t} p_{w_{i,j}} + \frac{\left(\frac{dS_w}{dp_{c_{ow}}} \right)_{i,j} p_{o_{i,j}}}{\Delta t} \right\}^p$$

$$+ \; S_{w_{i,j}} \beta_w \frac{T^{p}_{i,j} - T^{p-1}_{i,j}}{\Delta t}$$

where

$$\gamma = \frac{k \, k_{rw} \, \rho_w}{\mu_w \varepsilon} = \frac{k \cdot k_{rw}(S_w) \cdot \rho_w(p_w, T)}{\varepsilon \cdot \mu_w(T)}$$

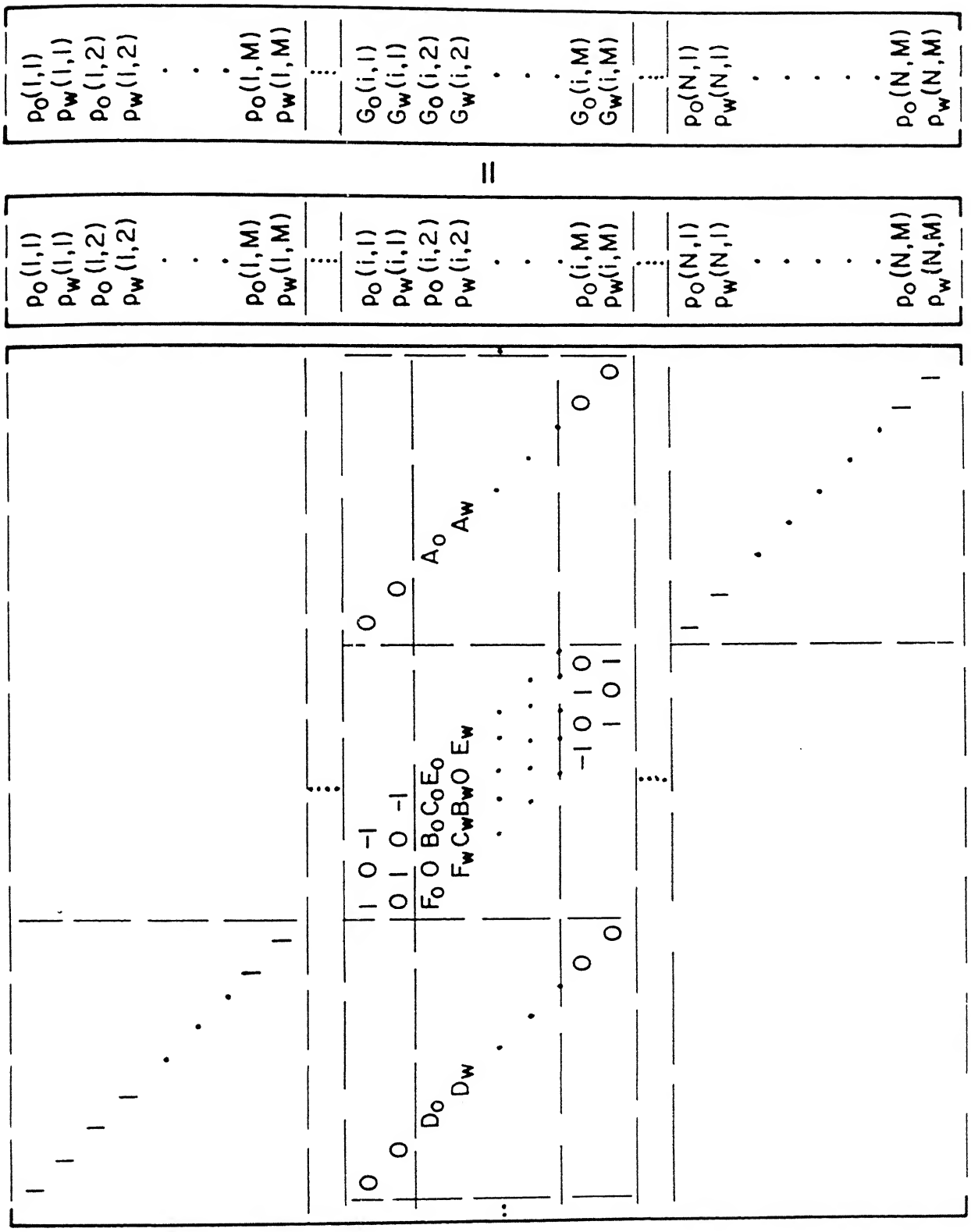
The discretized version of the boundary conditions at y=0, Y (See Section 4.8) are,

$$p_o(i,1) - p_o(i,2) = 0$$
$$p_w(i,1) - p_w(i,2) = 0$$
$$p_o(i,M-1) - p_o(i,M) = 0$$
$$p_w(i,M-1) - p_w(i,M) = 0$$

Dirichlet boundary conditions at x = 0, X are easily applied by replacing Equations(4.8,4.9) by the respective function values of the pressure.

4.10 Solution of the Pressure Equations

When the equations 4.8 and 4.9 of Sec. 4.9 are written for each node and such pairs of equations are drawn up for all the nodes in the domain {See Fig. 4.1}, a set of linear equations is obtained. The matrix structure is obtained:



The band width of the matrix is (2.(m-1)) and its size is (2 NM) x (2 NM). This matrix is inverted using a sparse matrix solver [29] that requires values of only those elements that lie within the band.

4.11 Solution of the Energy Equation

The energy equation is solved using the operator splitting algorithm.

The energy equation

$$\frac{\partial T}{\partial t} + U \frac{\partial T}{\partial x} + V \frac{\partial T}{\partial y} = \left(\frac{K_h}{\sigma_T}\right) \nabla^2 T$$ is split into the following steps,

Predictor: $$\frac{\partial T}{\partial t} + U \frac{\partial T}{\partial x} + V \frac{\partial T}{\partial y} = 0,$$

Corrector: $$\frac{\partial T}{\partial t} = \left(\frac{K_h}{\sigma_T}\right) \nabla^2 T ,$$

as described in Chapters 2 and 3.

4.11.1 Solution of the Predictor using Streamlines

We choose a curvilinear coordinate system ($\xi - \eta$) locally at each node such that $\nabla\xi$ is aligned with the net fluid composite velocity at that node, i.e. ξ is the streamline passing through the node {See Figure 4.19,(A)} ξ and η are orthogonal to each other. Then, it can be shown that

$$U T_x + V T_y = U' T_\xi \quad \text{where } U' = \sqrt{U^2 + V^2}$$

{See Appendix F(a)}

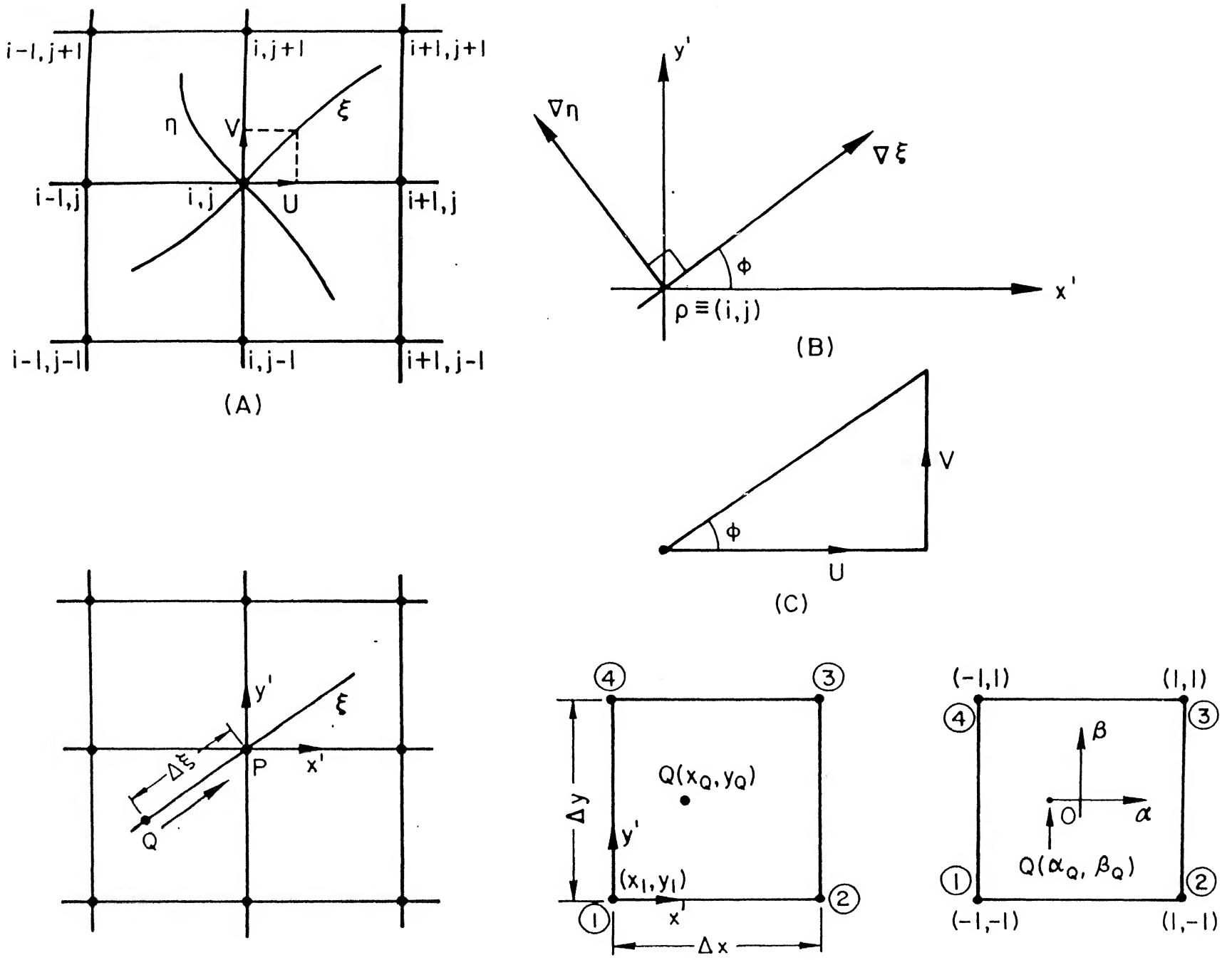


Figure 4.19 : Solution of predictor of energy equation.

Hence, the predictor step becomes,

$$T_t + U^* T_\xi = 0 \quad \text{where } U^* = U^*(x,y).$$

The solution of this equation is,

$$T(U^* t - \xi) = \text{constant}.$$

Since the grid surrounding each node is small, we assume the streamlines ξ to be straight, i.e.,

$$\xi = x^* \cos\phi + y^* \sin\phi \quad \{\text{See Figure 4.19, (D)}\}$$

and $\phi = \tan^{-1} \frac{V}{U}$

Here (x^*,y^*) are the local coordinates at (i,j) and (x,y) are the global coordinates for the grid.

The solution of the predictor step requires that the temperature at 'Q' at p^{th} timestep be transferred to the point 'P' at the $(p+1)^{th}$ timestep {See Figure 4.19, (D)}, i.e.,

$$T_p^{p+1}\,(U^*\,(t+\Delta t) - \xi) = T_Q^p\,(U^* t - (\xi - \Delta\xi))$$

where $\Delta\xi = U^{*p} \cdot \Delta t$.

See Appendix G for the details of interpolation required to determine T_Q.

The predictor step is implemented along the following lines,

(a) From U and V, calculate ϕ and $U^* = \sqrt{U^2 + V^2}$

(b) From the value of ϕ, find out the quadrant in which Q lies

(c) x_Q and y_Q are calculated using the following relations

$$\Delta\xi = U^* \Delta t$$

$$\Delta x_Q = \Delta\xi \cdot \cos\phi, \quad \Delta y_Q = \Delta\xi \cdot \sin\phi$$

$$x_Q = x_P - \Delta x_Q, \quad y_Q = y_P - \Delta y_Q$$

(d) T_Q is computed by interpolation and transferred to T_P

(e) Steps (a) - (d) are repeated for every internal node in the flow region.

N.B. At the boundaries, y = 0 and y = L, V ≡ 0

4.11.2 Solution of the Corrector Step using A.D.I.

The corrector equation is,

$$\frac{\partial T}{\partial t} = \left(\frac{K_h}{\sigma_T}\right)\nabla^2 T \qquad \{\text{from Section 4.11}\}$$

This equation is solved using ADI as described in Section 3.13 (Chapter 3). The only difference is that unlike Pe in Section 3.13, $\left(\frac{K_h}{\sigma_T}\right)$ is not a constant and therefore it needs to be calculated at each node.

4.12 Algorithm

The system of equations governing the distribution of oil and water pressures and temperatures are non-linear and mutually coupled. They must be solved simultaneously and by iteration.

The algorithm, in short, is as follows:

1. We start with the initial p_o, p_w, T and S_w values at t = 0.
2. The coefficients ($A_o - G_o$ and $A_w - G_w$) of pressure equations are computed using the values of p_o, p_w, T and S_w at the latest timestep.
3. A system of pressure equations, as shown in Section 4.10, is solved to obtain the new values of p_o and p_w.
4. S_w, ρ_o and ρ_w are updated using the new pressure values.

5. Darcy velocities v_{ox}, v_{oy}, v_{wx}, v_{wy} are calculated using the latest values of p_o, p_w, S_w, ρ_o and ρ_w. (See Appendix G).

6. Using S_w, ρ_o, ρ_w , $\tilde{v}_o$, $\tilde{v}_w$, the coefficients of the energy equation are computed and the equation is solved using the OS algorithm yielding a new temperature T.

7. ρ_o and ρ_w are updated once again using the new T.

8. Steps 2 to 7 are repeated until convergence of p_o, p_w and T is achieved.

9. Fresh computation is initiated for the next time level starting from step 2.

4.13 Results and Conclusion

The data for oil and water properties and those of the porous formation used in this problem is given in Tables 3 and 4. This data is adapted from the data used by Boberg [17], for a laboratory test which reproduced steam and hot water injection processes on a small scale.

The domain which is used for calculating ppv and for plotting the scalars has a length X_d (= 1.6 m) which is smaller than the overall domain length X (= 4 m) {See Fig. 4.1}. In this manner, the influence of the right hand boundary conditions is delayed. On a smaller domain, in the isothermal case, there is a build up of oscillations in the saturation profile. For a total domain size of 4 m, oscillations do not appear for a period of 1 hour of simulation and the flushing effect of water on oil is seen over a distance of 1m.

The computer program developed in the course of the present

study showed the following performance. On Convex Minisuper computer, the isothermal version of the program took two minutes of CPU time for a 41x5 grid and a time step Δt = 0.01 hour. The non-isothermal version took three CPU minutes. Convergence criteria used was $(Q^{P+1} - Q^{P})/Q^{P+1} \times 100 < 0.01$ where Q is p_o, p_w and T. The convergence was achieved, on an average, after five iterations. In order to minimise the computational instabilities, smoothened initial conditions were used (See Section 4.8). Further, timestep was gradually increased from 0.1 (Δt) to Δt, where Δt is the characteristic timestep.

The correctness of the computer code and the matrix inverted has been tested by solving the linear transient heat conduction equation.

Now, we would be presenting the results in a format given in Section 4.6.

(i) Isothermal problem: Figure 4.2 shows the rise in saturation level of water in the flow domain along its centreline with time. There is no front and the profile is diffusion dominated. Next, we consider the effect of raising the formation temperature. As we raise T_f from $30^{o}C$ to $50^{o}C$, we find an increase in the water saturation along the centreline of the flow domain (See Figure 4.3). We also observe in Figure 4.4, that oil recovery improves with the raising of the formation temperature. This is expected because of the drop in oil viscosity with an increase in temperature. Evolutions of centreline velocity profiles of water and oil are shown in Figures 4.5 and 4.6 respectively.

(ii) Oil displacement efficiencies of the isothermal (cold water injection) and non-isothermal (hot water injection) cases are

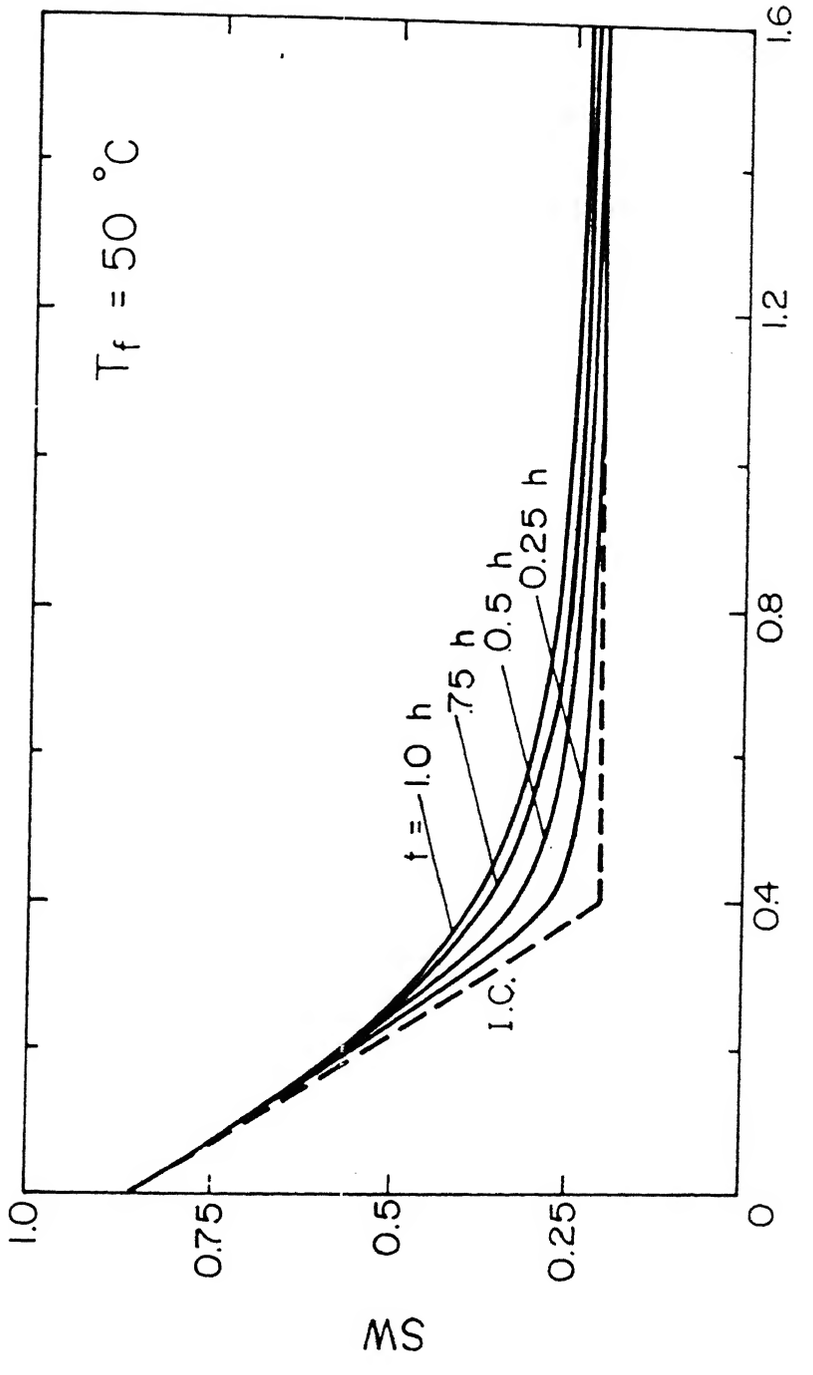


Figure 4.2 : Isothermal case : Progress of saturation front.

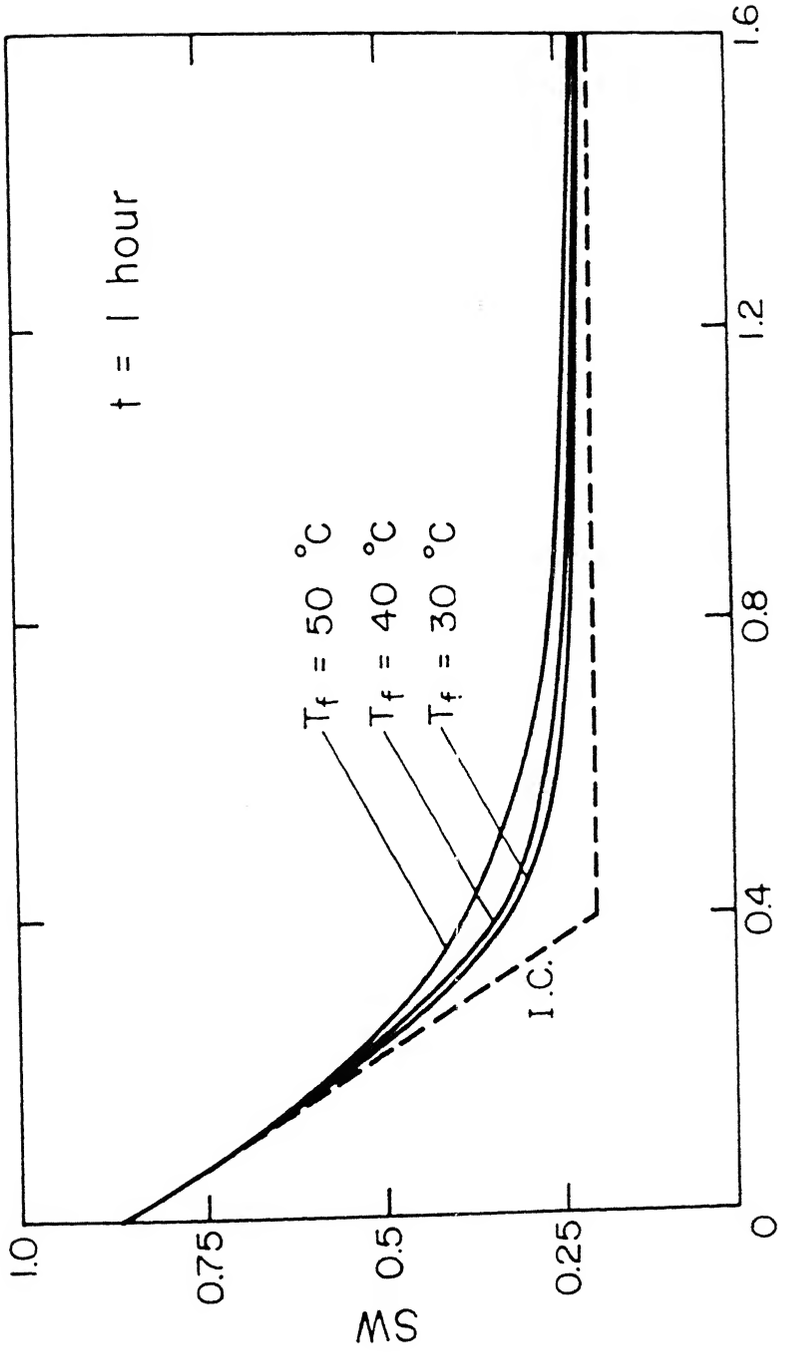


Figure 4.3 : Isothermal case : Effect of formation temperatures (T_f) on saturation front.

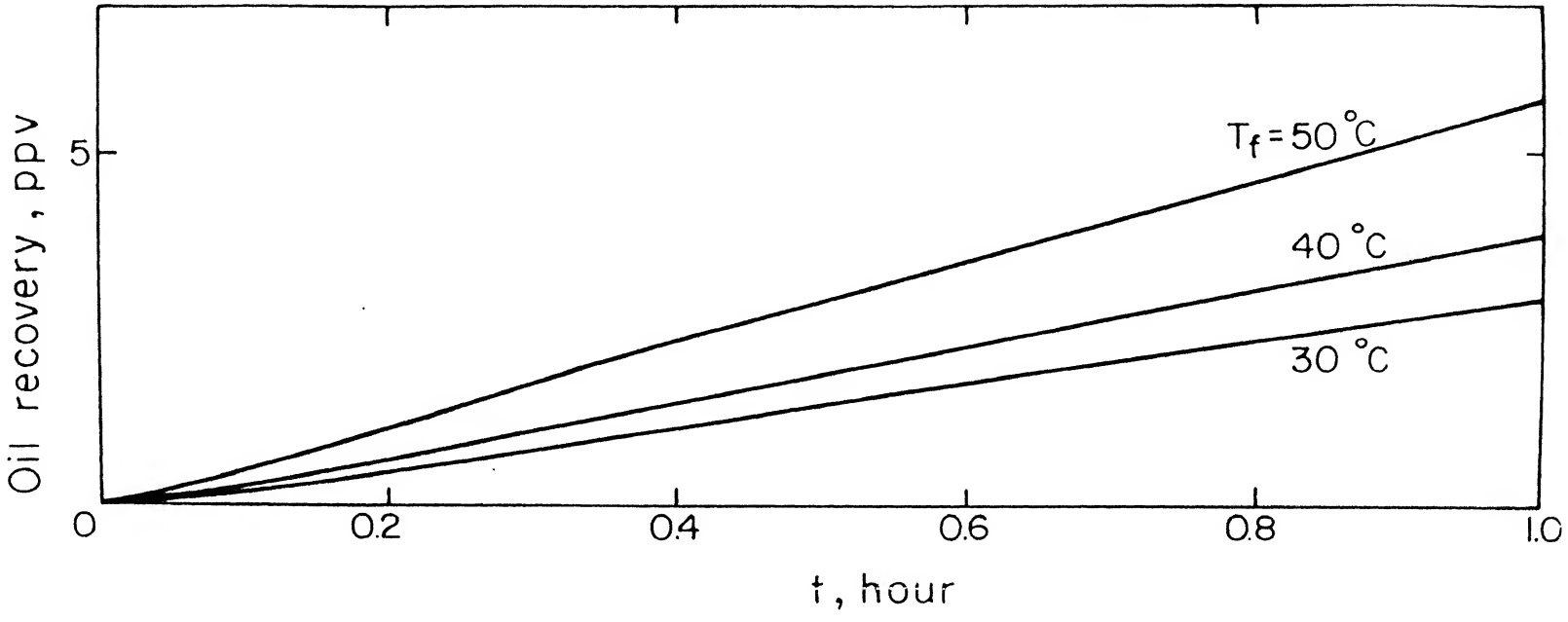


Figure 4.4 : Isothermal case : Effect of formation temperature (T_f) on oil displacement efficiency.

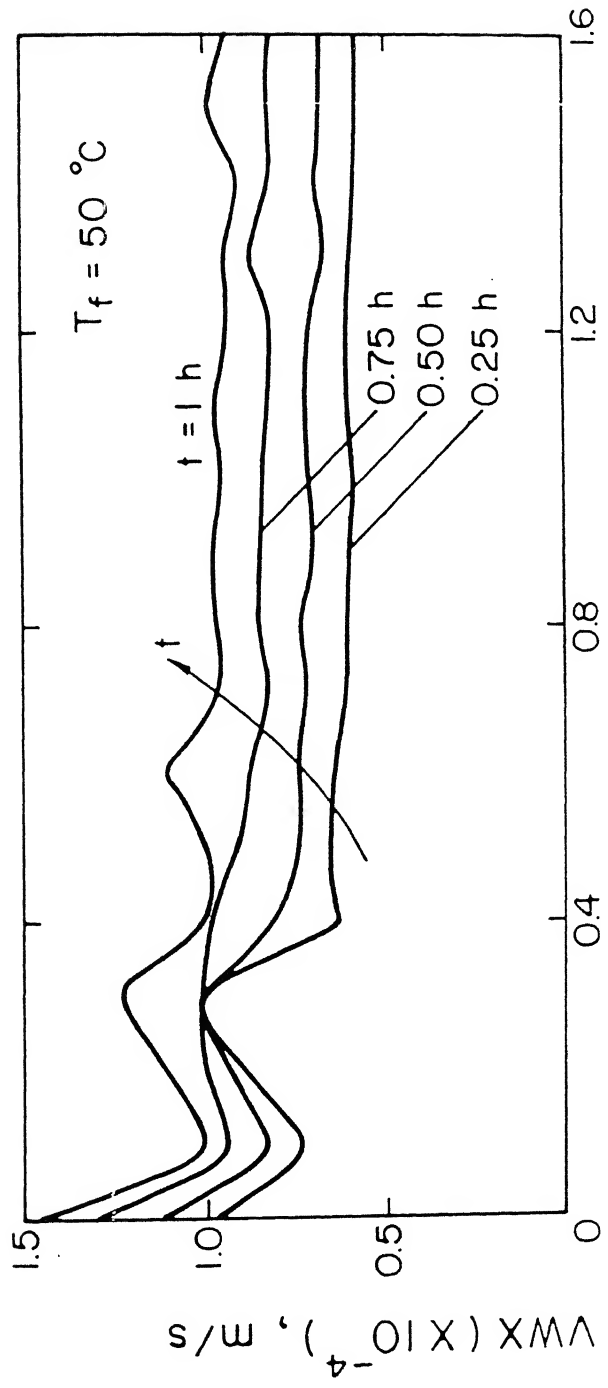


Figure 4.5 : Isothermal case : Change in water velocity profile with time.

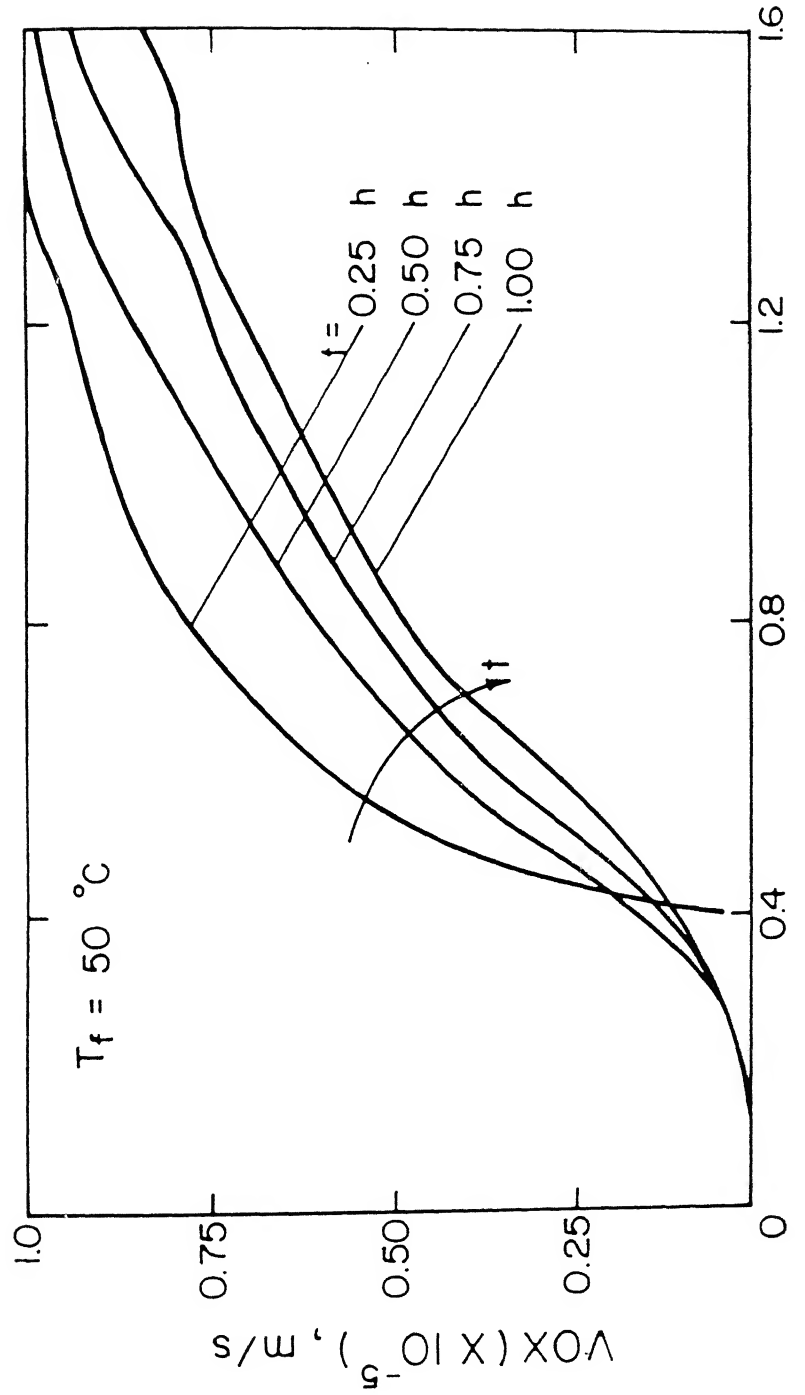


Figure 4.6 : Isothermal case : Change in oil velocity profile with time.

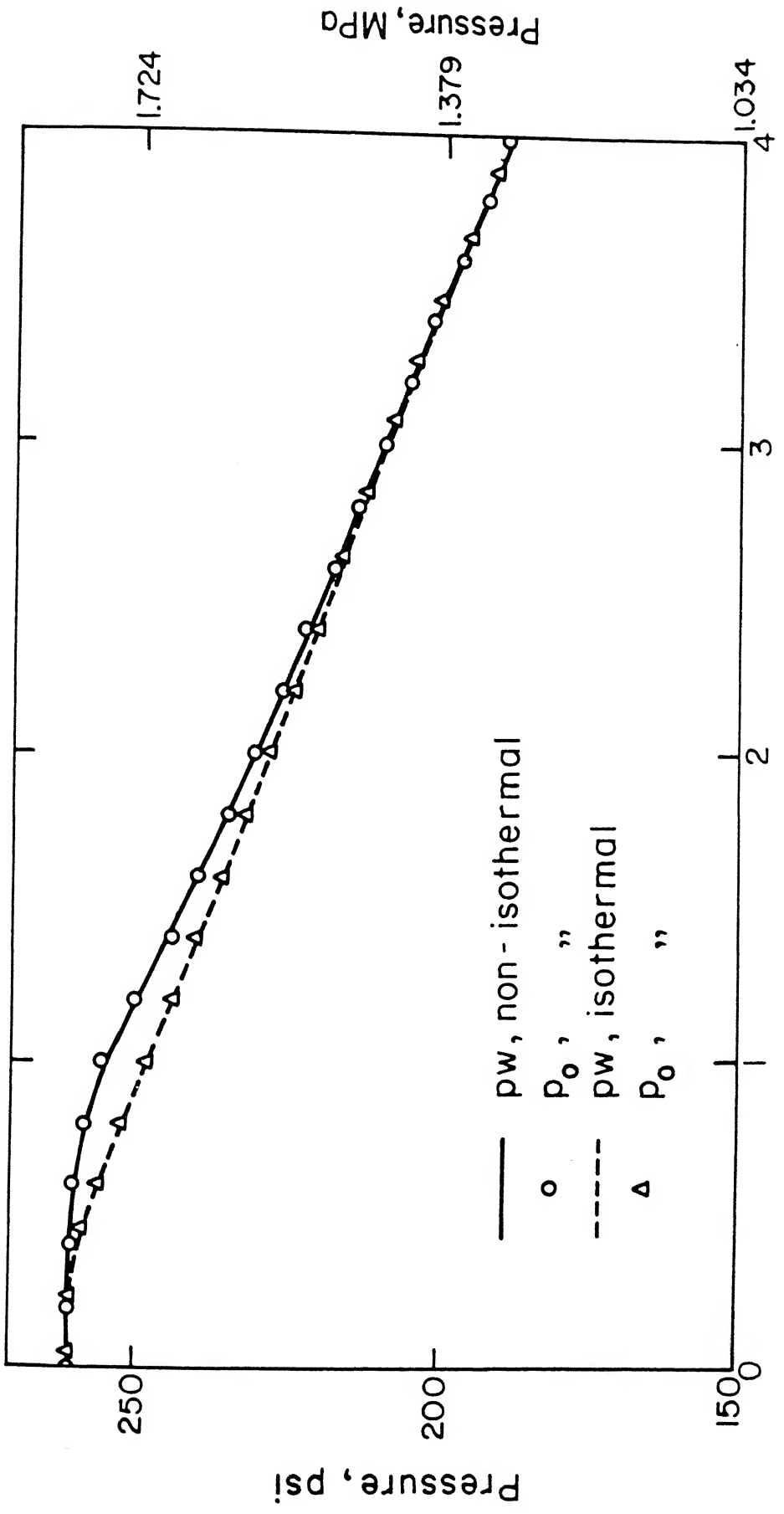


Figure 4.7 : Non-isothermal case : pressure profiles

compared in Figure 4.8. The percentage pore volume (ppv) of the non-isothermal case is below that of the isothermal case for times less than 1 hour. Beyond one hour, the performance of the non-isothermal process is better than that of the isothermal process. A possible explanation for this phenomena is given in part (iii) below.

(iii) We analyze next the hot water injection technique in detail. Figures 4.9 and 4.10 show the development of centreline saturation and temperature profiles. Both of them show a distinct front like structure and on comparing them, we observe that saturation front moves with the same speed as the temperature front. The saturation profile reveals an unusual feature namely a dip in S_w immediately beyond the front. This dip is responsible for the initial low level of the ppv of the non-isothermal case as compared to the isothermal case (See Figure 4.8).

Figures 4.11 and 4.12 give the development of water and oil velocity profiles respectively for the non-isothermal case. We find that the magnitude of velocities is higher than that of the isothermal case. The profiles individually show the presence of a peak which travels with the same speed as the two aforementioned fronts.

(iv) We now study the effect of heat loss to overhang and underhang on oil recovery. As mentioned earlier, (Section4.5) this is effected through the manipulation of the parameter Bi. Higher the value of Bi, greater the heat loss. Since this is convection dominated transport phenomena (See Appendix B(b)),

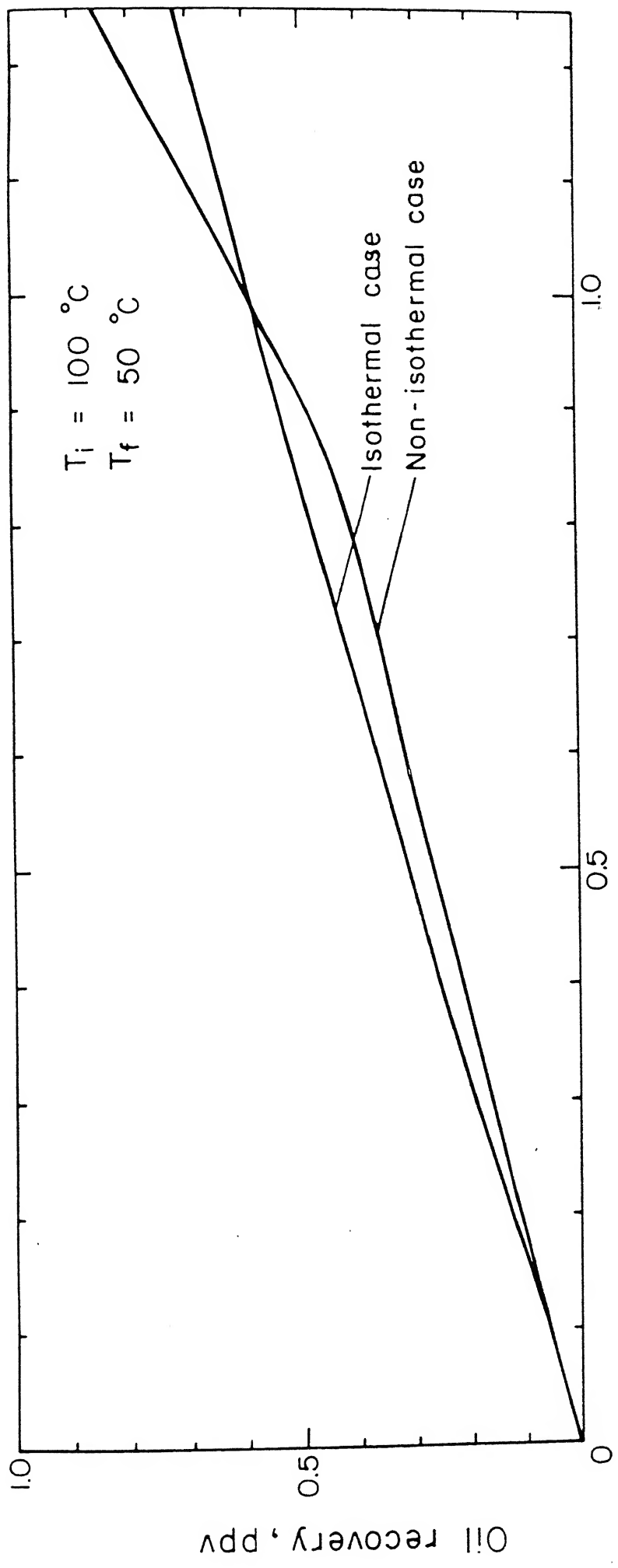


Figure 4.8 : Comparison of oil displacement efficiencies of isothermal and non isothermal cases.

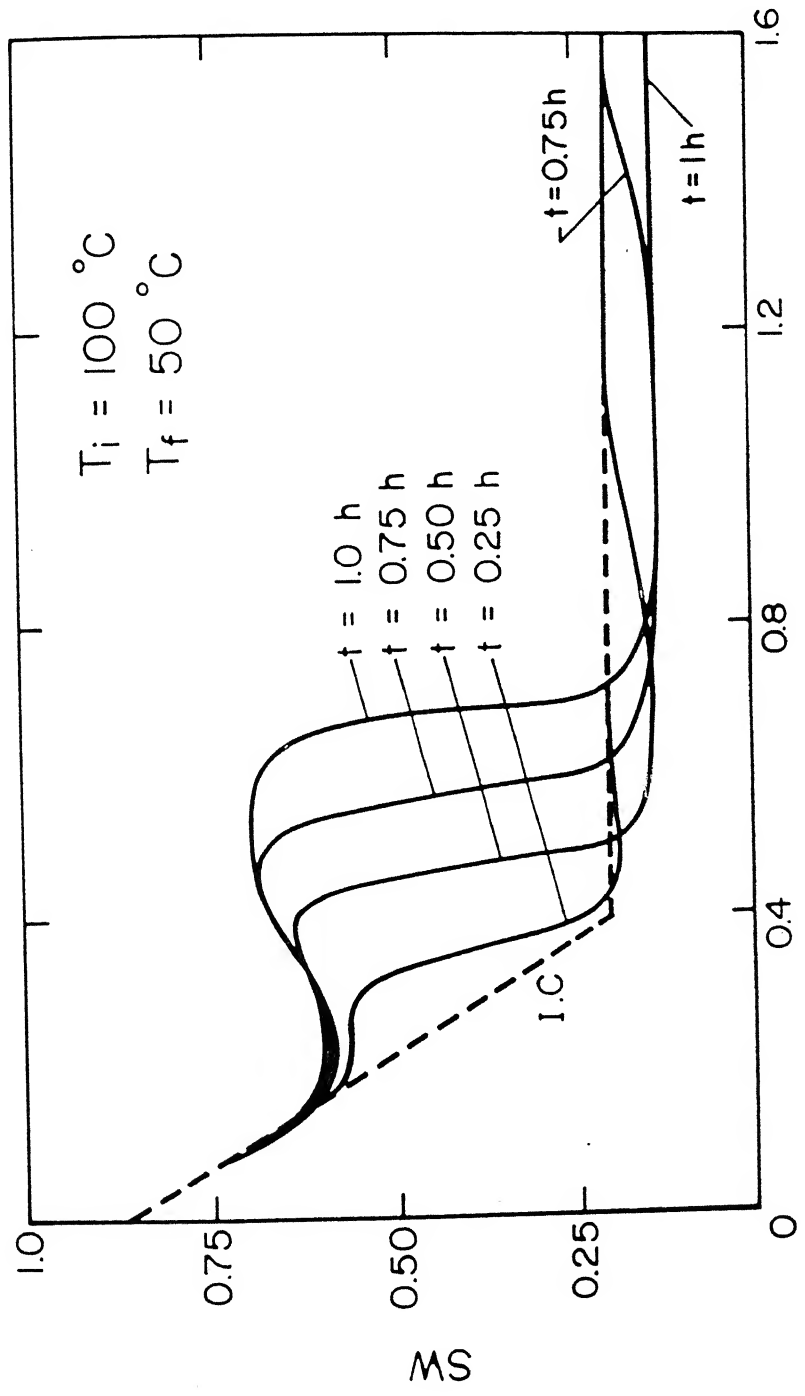


Figure 4.9 : Non isothermal case : Progress of saturation front.

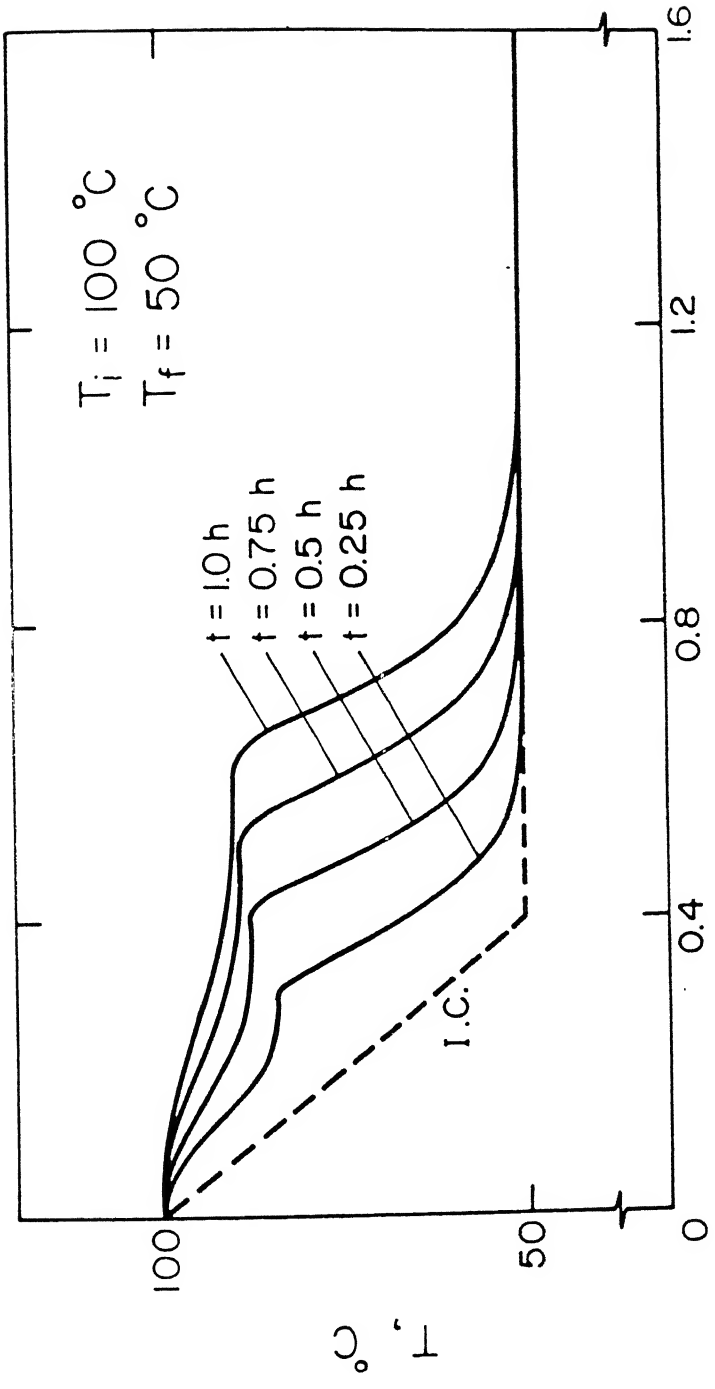


Figure 4.10 : Non isothermal case : Progress of temperature front.

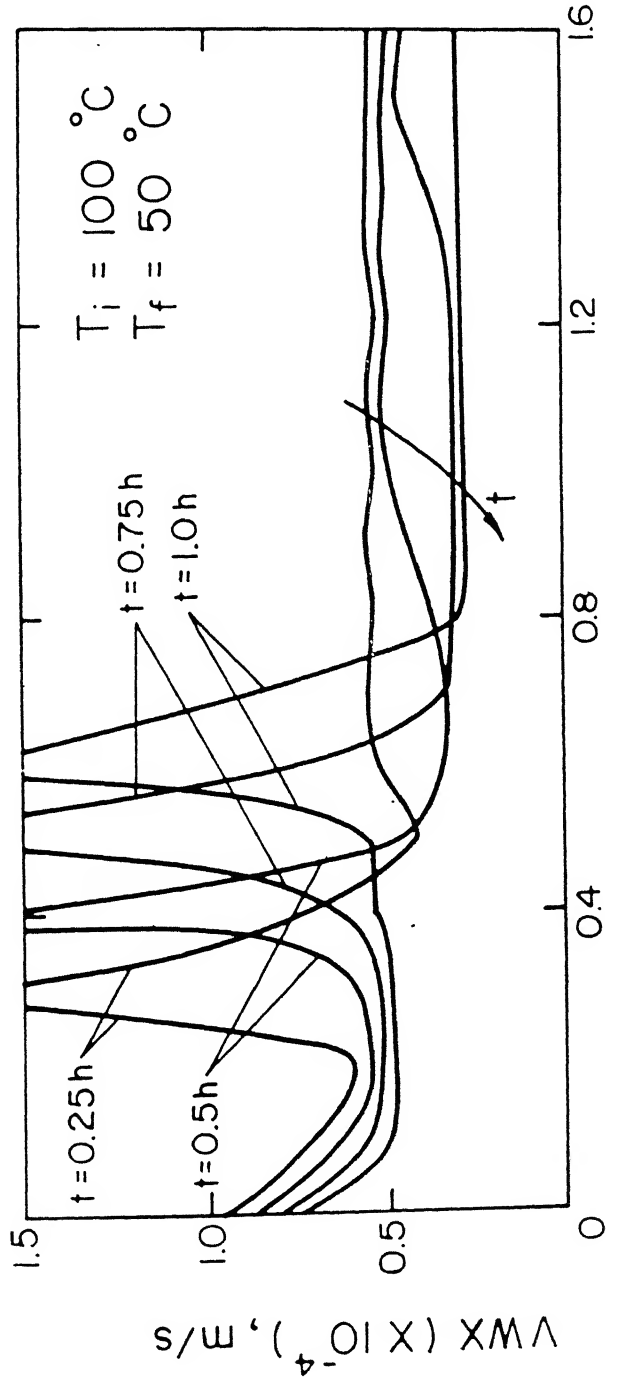


Figure 4.11 : Non isothermal case : Changes in water velocity profile with time.

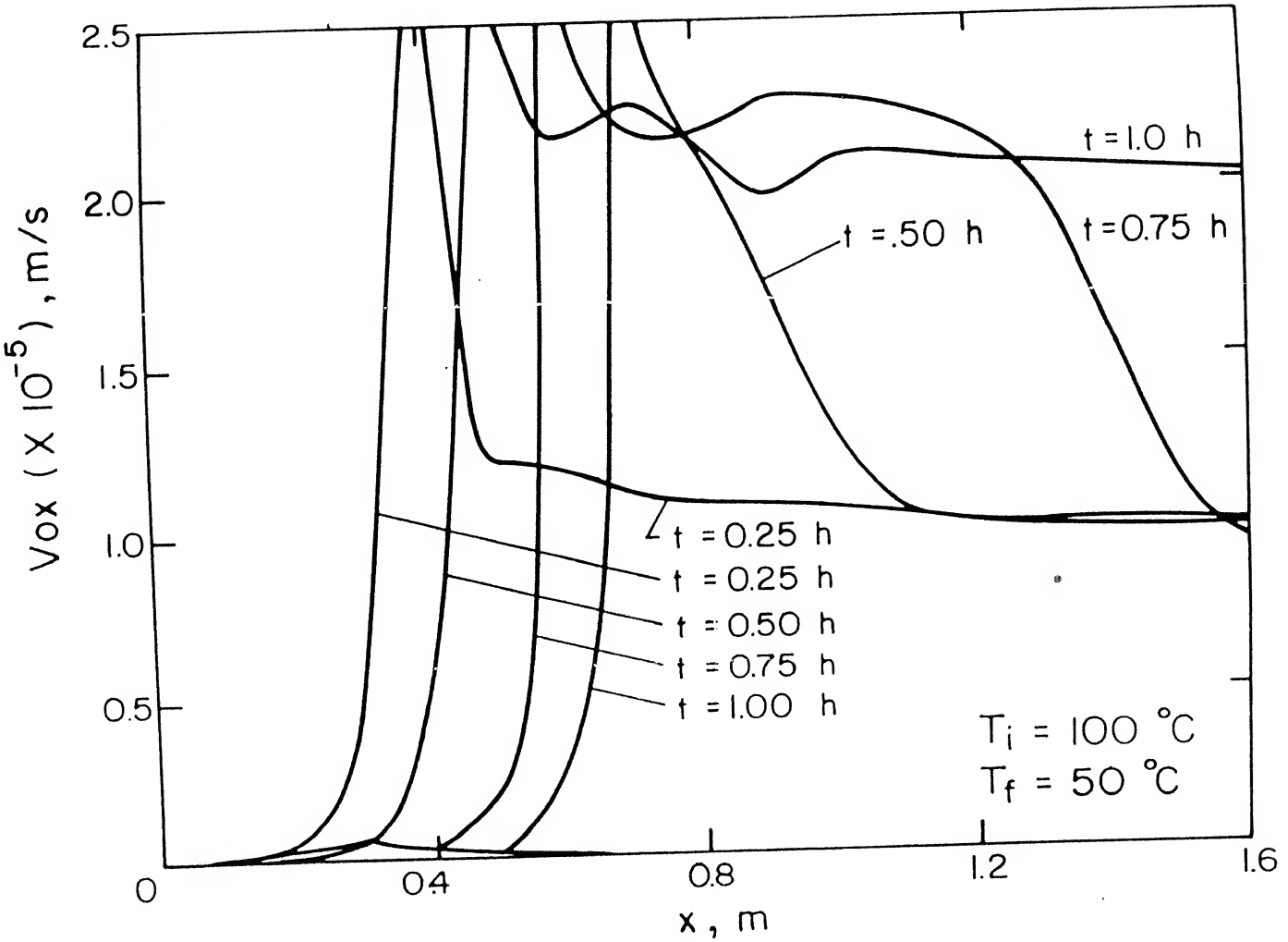


Figure 4.12 : Non isothermal case : Changes in oil velocity profile with time.

for the time frame considered (~ 1 hour), the effect of heat loss does not penetrate deep enough to influence the centreline characteristics significantly for the domain width considered (Y = 0.8m). At smaller widths (Y = 0.04 m, See table 3) the influence of heat loss is seen to be significant.

Figure 4.13 shows the effect of increase of Bi on centreline saturation profile. As the heat loss increases (i.e. Bi increases), the saturation front profile becomes increasingly smooth. Study of the centreline temperature (Figure 4.14) reveals the cause of the decrease. As Bi is raised, a smaller amount of thermal energy is available causing the weakening of the centreline temperature front. This leads to a higher average viscosity and so a lower fluid mobility. Hence the volume of water being pushed through the sand is reduced, and consequently, water saturation profile is lowered.

In Figure 4.15, the typical widthwise temperature profiles for various Bi values are shown. When Bi = 0.01 and 1, the profile is a straight line which implies that the overhang and the underhang are perfect insulators. But as Bi is raised to 100 and then to 10,000, the profile increasingly becomes parabolic in nature. The slope is a maximum at the top and the bottom of the domain because of the heat loss to the rocks.

The effect of heat loss on ppv and the oil displacement efficiency is shown in Figure 4.16. We observe that for $t < 1.5$ hour, oil recovery (ppv) for Bi = 0.01, 1, 100 and 10,000 are nearly equal. At the end of 2 hours, a clear trend can be discerned. We find that oil recovery is the least for Bi =

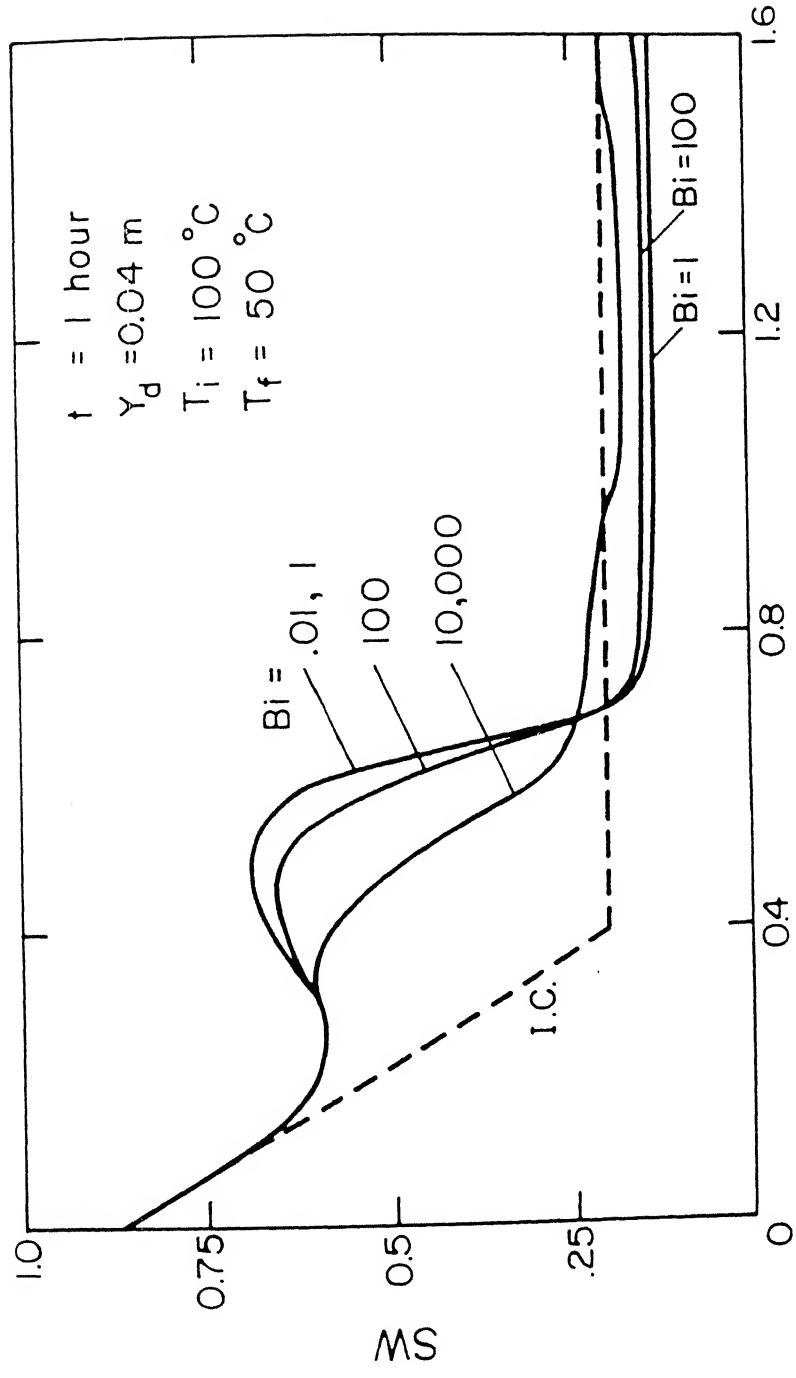


Figure 4.13 : Effect of Bi on saturation front.

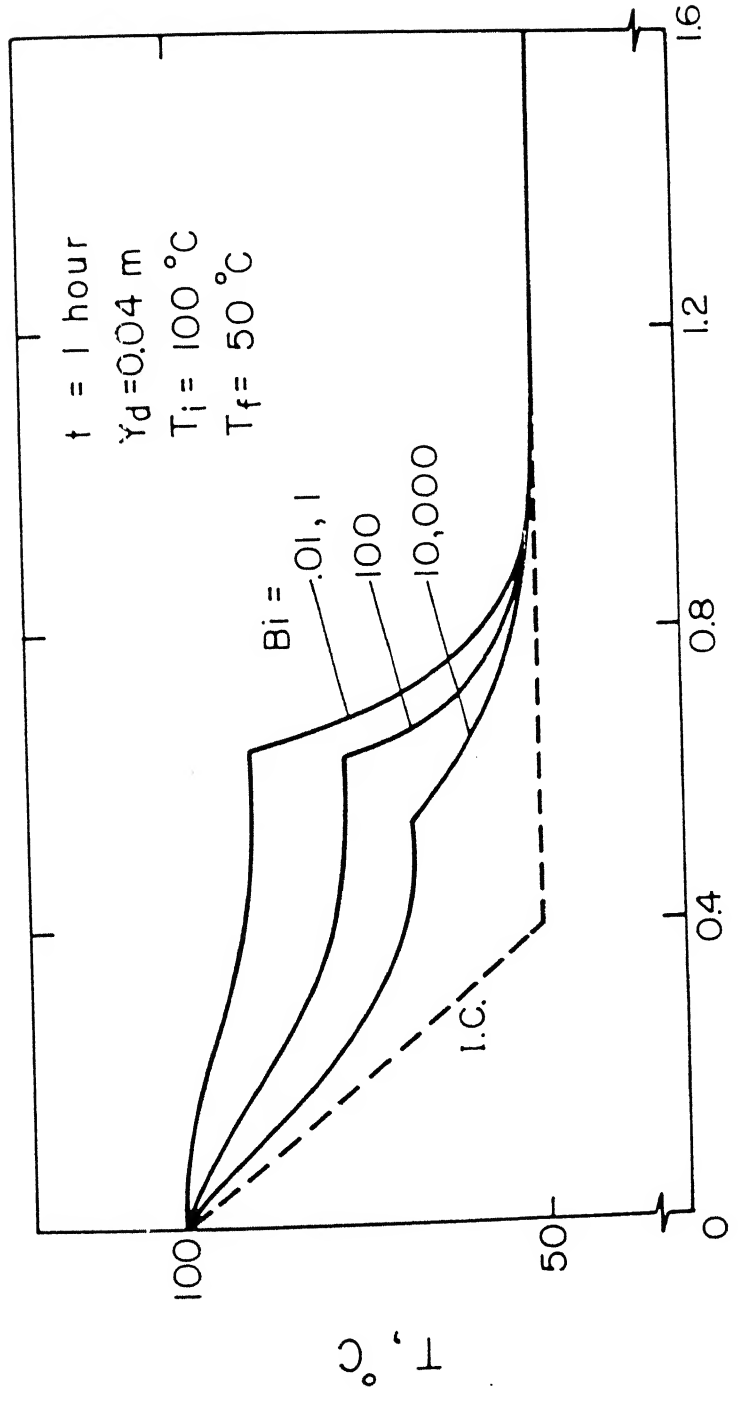


Fig. 4.14 : Effect of Bi on temperature front.

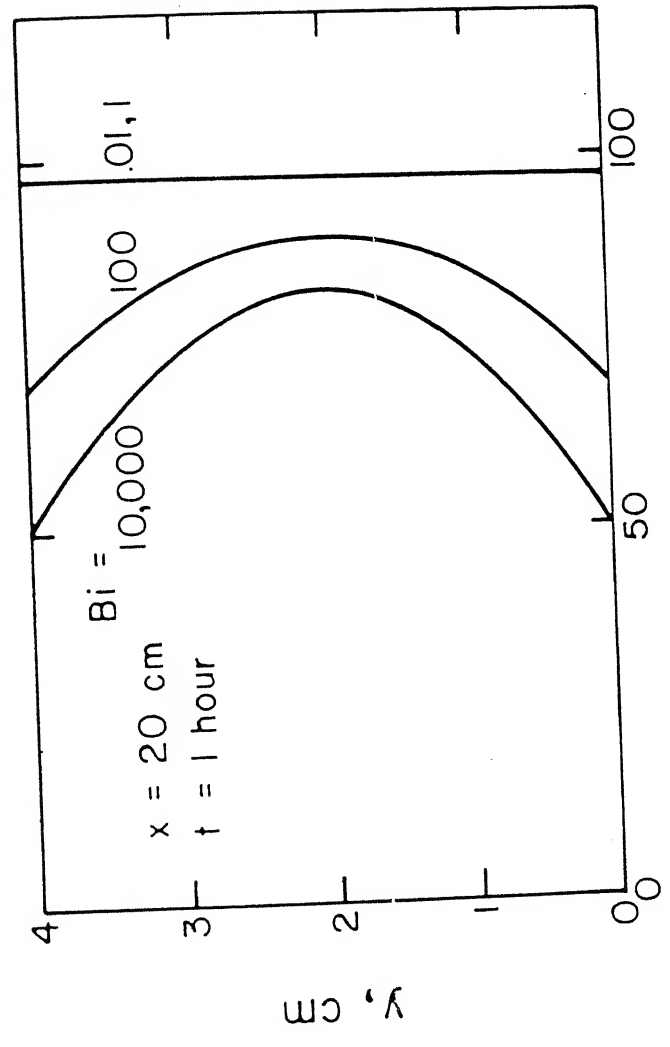


Figure 4.15 : Effect of Bi on widthwise temperature profile.

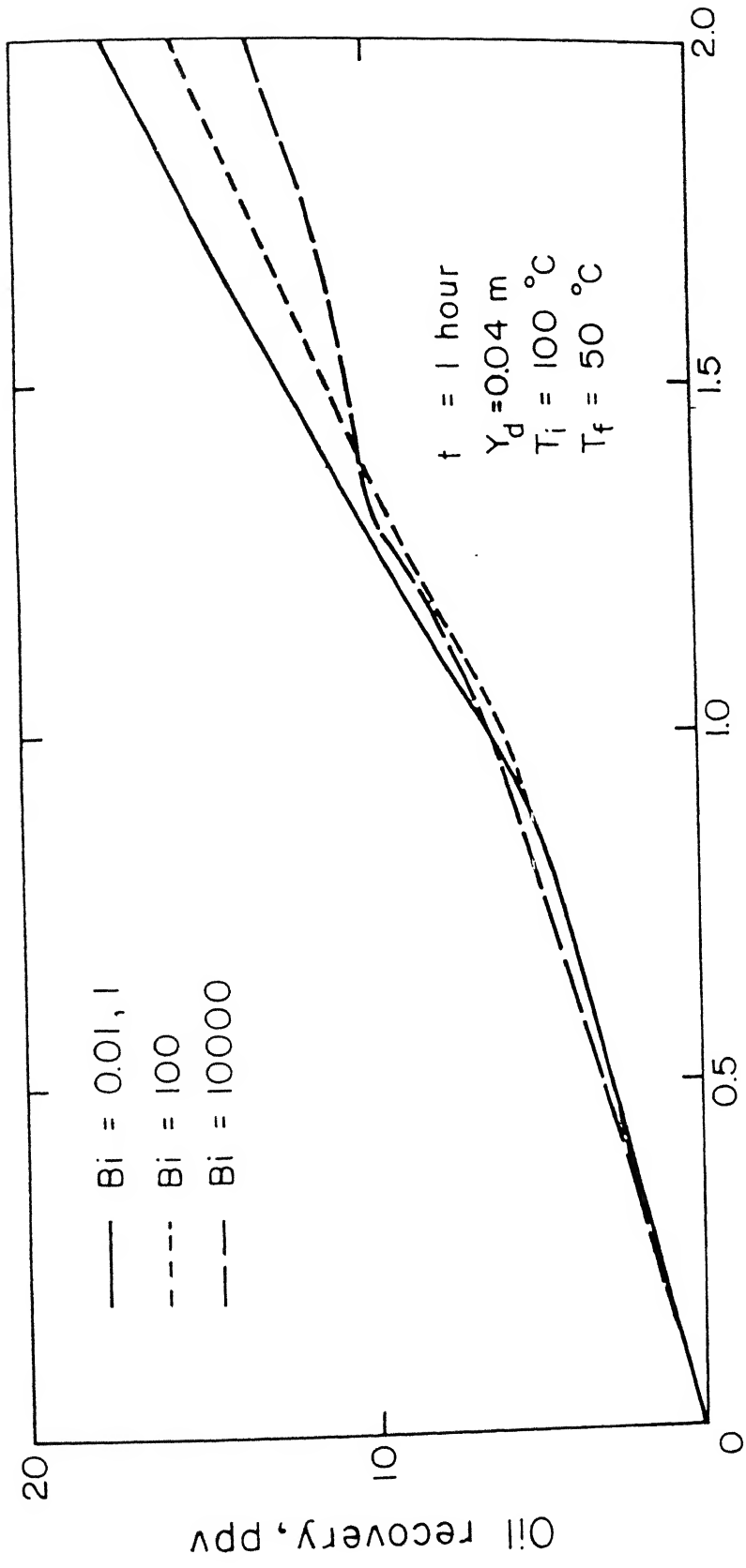


Figure 4.16 : Effect of Bi on oil displacement efficiency.

10,000 and a maximum for Bi = 0.01 and 1. Hence we conclude that in the long run, heat loss from the sand bank during hot water injection adversely affects the oil production.

We now discuss the effect of the size of the grid on resolution of fronts of various scalars. In Figure 4.17, the consequence of progressive grid coarsening (41x5, 31x5 and 21x5) on the centreline S_w profile is studied. The difference in the profiles is striking for the cases 31x5 and 21x5, but the differences become smaller as we move on to a 41x5 grid. We observe that it becomes increasingly difficult to keep track of the saturation front faithfully with grid coarsening. Not only does the front lose its true shape, its speed also becomes erratic. Same conclusions can be derived from the study of centreline temperature profiles (Figure 4.18). Results presented in this chapter use a 41x5 grid.

Finally, the role of the field data in the form of $K_{ro}(S_w)$ and $K_{rw}(S_w)$ and $p_{c_{ow}}(S_w)$ (Table 4) must be mentioned. Here, $k_{ro,w}$ is given for a finite set of S_w values. A linear interpolation is used to compute intermediate values of the relative permeability and capillary pressure. Recent literature (Reference [25]) shows that the prediction of a numerical simulator for flow through a porous media is sensitive to the values of K_{ro}, K_{rw} and p_c. Any deviation from the correct value of these inputs can lead to unrealistic predictions. It is suggested here that closed form analytical expression be used to furnish the relative permeability and capillary pressure values as in Reference [27]. These are expected to improve the quality

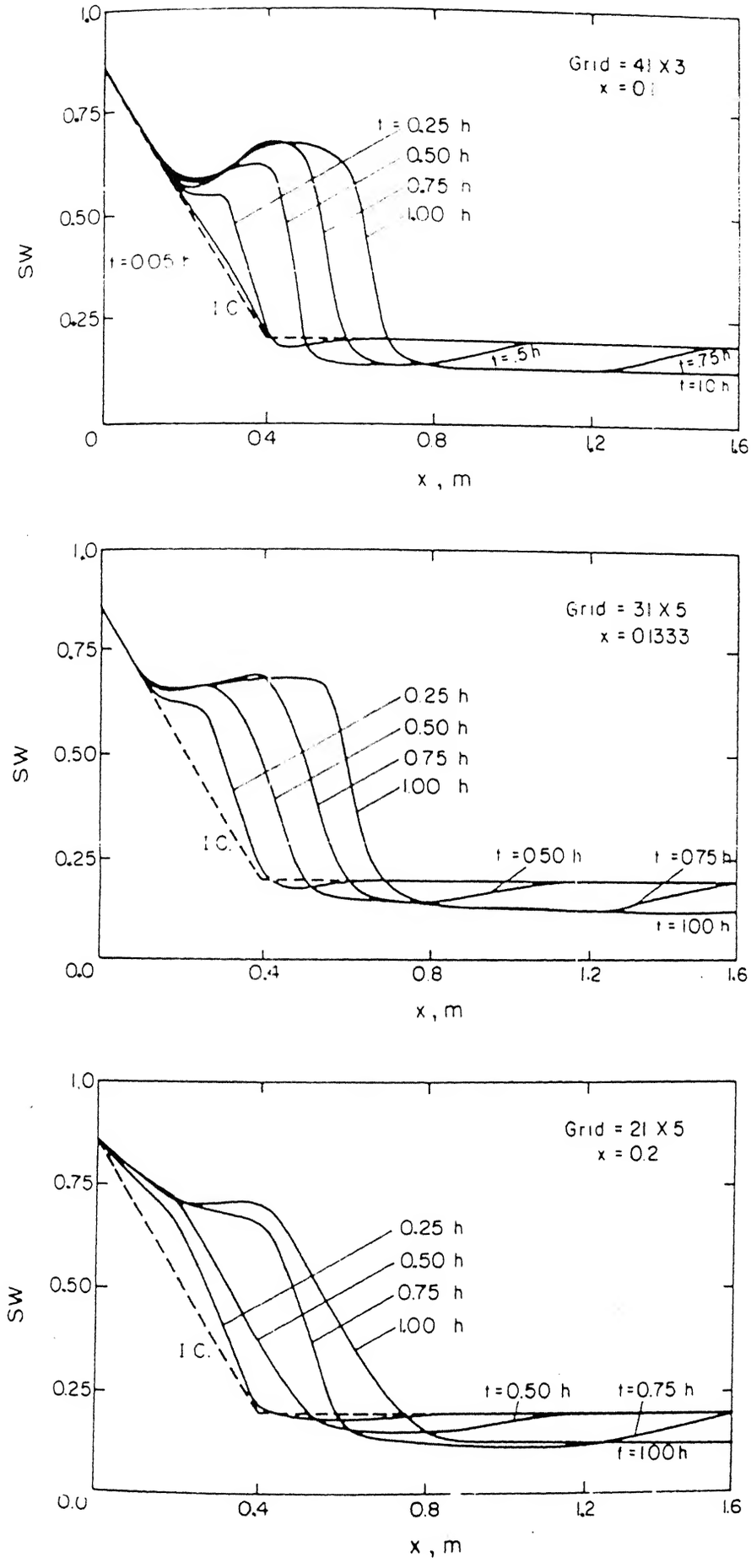


Figure 4.17 : Effect of grid size on water saturation front progress : non-isothermal case.

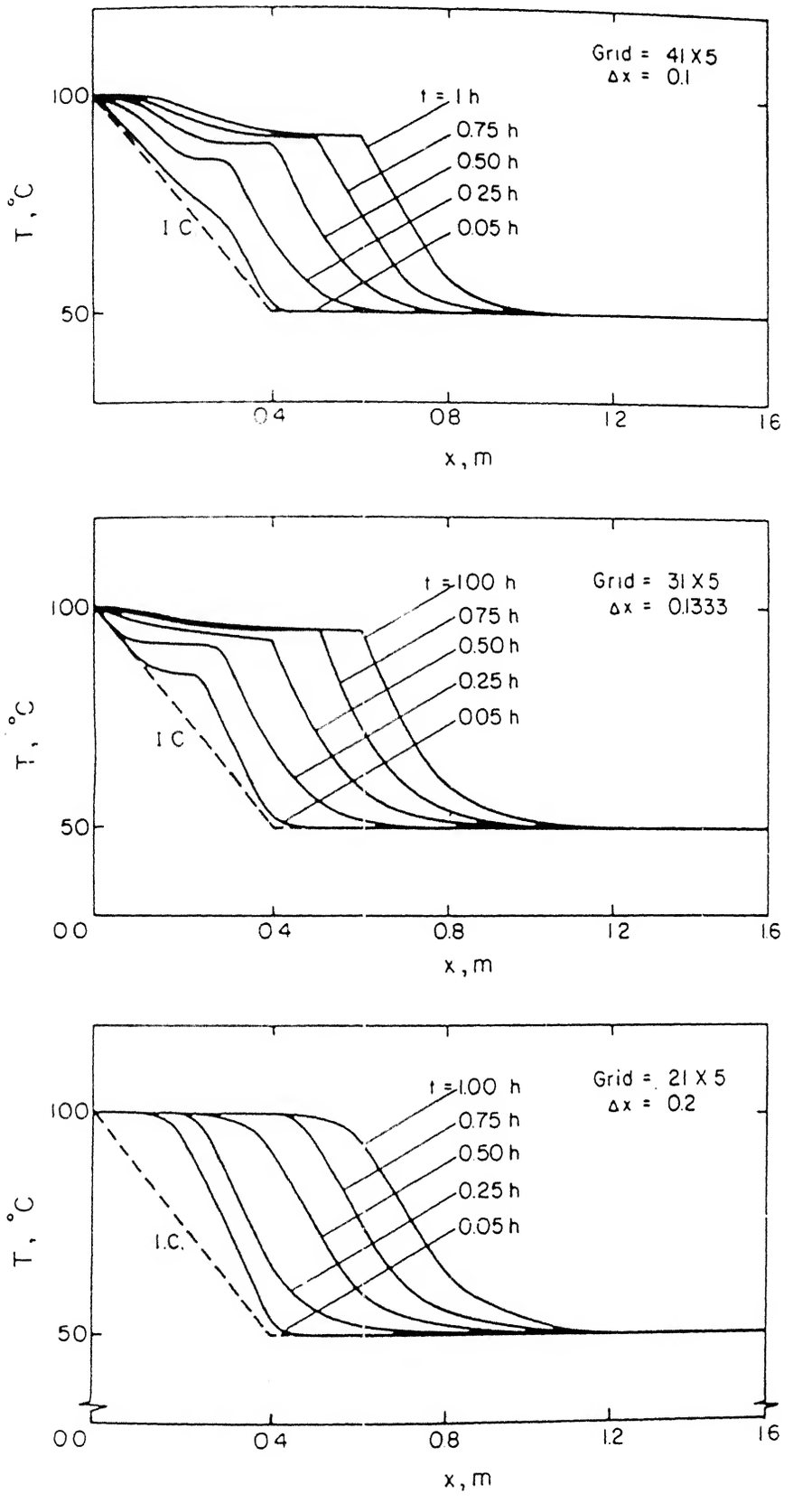


Figure 4.18 : Effect of grid size on temperature front progress : non-isothermal case.

of the numerical predictions.

Before closing this chapter, we would like to mention the advantages of harmonic averaging over the arithmetic one of θ and γ values (See Section 4.9). The reason behind the choice of harmonic average is that viscosity has very sensitive exponential dependence on temperature and the relative permeability changes sharply with saturation. This can cause sharp variations in θ and γ values in a small region. The choice proved to be a correct one as the wiggles in saturation profiles in the isothermal case were reduced in magnitude or not present at all.

Table 3 : Data Used in the Problem : Parameter Values

Most of the values in part (B) are taken from p224, Boberg [17].

(A) Domain length, X = 4.0 m

Domain width, Y = 0.8 m

Domain width, Y = 0.04 m (To study Bi effect)

Domain grid size, Δx = 0.1 m

Domain grid size, Δy = 0.2 m

Domain grid size, Δy = 0.01 m (To study Bi effect)

Domain of analysis ('problem domain' : Figure 4.1),

X_d = 1.6 m

time step, Δt = 36 s

(B) Permeability, k = 132 Darcies

Porosity, ε = 0.375

Compressibility: Oil, $\xi_o = 5 \times 10^{-6}$ 1/psi [=0.03447 1/Pa]

Compressibility: water, $\xi_w = 3.1 \times 10^{-6}$ 1/psi [=0.02137 1/Pa]

Expansivity: Oil, β_o = 0.00041 $1/^oF$ [$=2.28 \times 10^{-4}$ $1/^oC$]

Expansivity: water, β_w = 0.00041 $1/^oF$ [$= 2.28 \times 10^{-4}$ $1/^oC$]

Specific heat: oil, C_o = 0.5 Btu/lbm-^oF [=2092.0 W-s/kg-^oC]

Specific heat: water, c_w = 1 Btu/lbm-^oF [= 4184.0 W-s/kg-^oC]

Heat capacity/volume of the porous medium,

$(\rho c)_R$ = 36 Btu/ft^3-^oF [= 2412996.5 W-s/m^3-^oC]

Thermal conductivity of the porous medium,

K_h = 40 Btu/ft-day-^oF [= 0.1661 W/m-^oC]

Injection temperature, T_i = 100^{o}C

Formation temperature, T_f = 50^{o}C

Injection pressure (water) = 260 psi [= 17.926x10^{5} Pa]

In-situ pressure (water) = 190 psi [= 13.1x10^{5} Pa]

Table 4 : Data Used in the Problem : $\mu_w(T)$, $\mu_o(T)$, $K_{rw}(S_w)$, $K_{ro}(S_w)$, $p_{c_{ow}}(S_w)$ Tables

(i) $\mu_o(T)$:

$T(^oF)$	μ_o(cp)
50	5000
150	126
250	20
350	7.1
450	4.0

(i) $\mu_w(T)$

$T(^oF)$	μ_w (cp)
60	1.130
75	0.935
80	0.875
100	0.685
120	0.560
150	0.430
200	0.308
250	0.230
300	0.182
400	0.145
500	0.120

(iii) $k_{rw}(S_w)$, $k_{ro}(S_w)$, $p_{c_{ow}}(S_w)$

S_w	k_{rw}	k_{ro}	$p_{c_{ow}}$ (psi)
0.1	0	1.0	4.111
0.2	0.0016	0.875	0.095
0.3	0.0081	0.735	0.072
0.4	0.0259	0.590	0.061
0.5	0.0672	0.420	0.051
0.6	0.1000	0.210	0.041
0.7	0.1400	0.070	0.031
0.8	0.2000	0.016	0.021
0.86	0.2500	0	0.011

For intermediate values, linear interpolation was used to calculate oil and water viscosities (μ_o and μ_w), oil and water relative permeabilities (k_{ro} and k_{rw}). While computing saturation from capillary pressure $p_{c_{ow}}$, for $p_{c_{ow}} \geq 0.95$ psi, exponential interpolation was used. For $p_{c_{ow}} < 0.95$, linear interpolation was used.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

The performance of operator splitting algorithm was tested against those of the upwind and central difference methods by solving simple one and two dimensional benchmark problems. Results were compared against the analytical solution of problems which were available in some cases. It was found that for low Peclet number flows, the performance of OS was identical to the performance of the other two schemes and it matched the analytical solution (whereever it was available). For high Pe cases only the results by the OS came close to the analytical solution. Upwinding showed egregious false diffusion errors and central differencing was also not satisfactory. Hence we have concluded that the new scheme is relatively free from the basic errors afflicting the conventional lower order schemes namely false diffusion and grid orientation errors. OS was also successful in tracking fronts in high Pe flows.

The new technique was successfully used in modelling hot water injection technique for enhanced oil recovery and the effects of various parameters on oil recovery were demonstrated. They were as follows:

- improvement in oil production by raising the formation temperature T_f.
- superiority of hot water injection method over the cold water approach.
- adverse effect on oil production of heat loss to the surrounding rocks.

There is one feature of the new algorithm which needs improvement. We have solved the hyperbolic part of the convection-diffusion equation analytically using the formula $T(x, t + \Delta t) = T(x - \Delta t, t)$. If $\Delta t = \Delta x$, we obtain $T^{p+1}_{(x)}$ by equating it to $T^{p}_{(x-\Delta x)}$. If $\Delta t = \Delta x$, interpolation between $T^{p}_{(x)}$ and $T^{p}_{(x-\Delta x)}$ is used. Interpolation on a coarse mesh causes errors which are similar to false diffusion (See Figure 5.1). As shown in the figure smaller the value of $\Delta t/\Delta x$, higher the error. A numerical scheme could have been used to solve the hyperbolic part, but the advantages of accuracy of the analytical solution would then have been lost. Figure 5.1 shows that interpolation errors diminish on a refined mesh.

Scope for Future Work

Before declaring the operator-splitting method as an efficient alternative to the upwind scheme, it needs to be tested on various other flow configurations. Some of these are convection-diffusion equations with source term and radioactive decay and to fluid flow problem. In the context of oil recovery, the OS algorithm should be compared with those which are already in use in the industry.

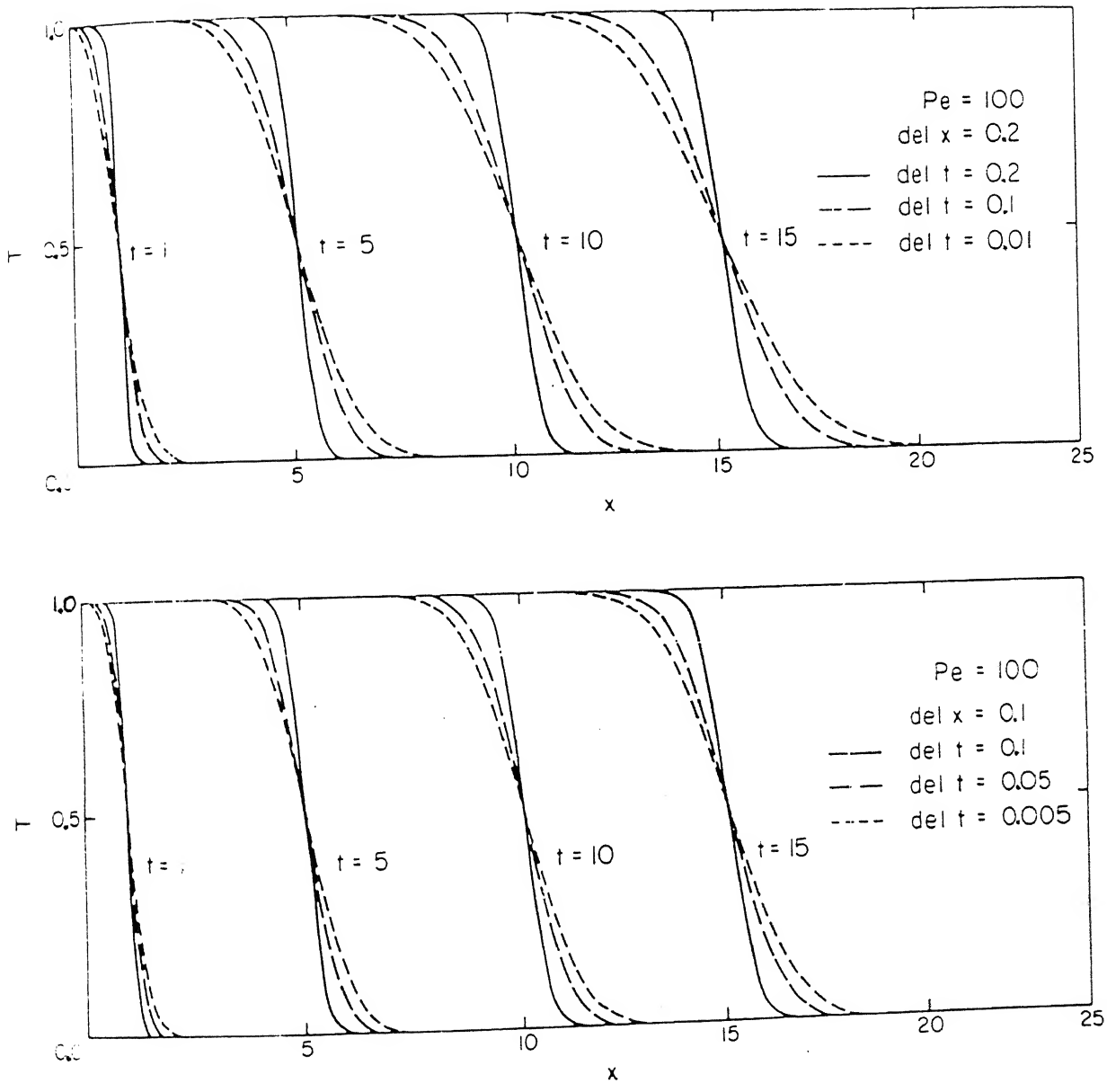


Figure 5.1 : Effect of $\Delta t/\Delta x$ on front resolution by operator splitting (High Pe one dimensional case)

REFERENCES

(1) Leonard, B.P., 1979, "A survey of finite differences of opinion on Numerical muddling of the incomprehensible defective confusion equation", Editor TJR Hughes, Finite Element Method in Convection dominated Problems, ASME Publications, New York.

(2) Leonard, B.P., 1979, "A stable and accurate convective modelling procedure based on quadratic upstream interpolation", Computer methods in applied mechanics and engineering, Vol. 19, pp. 59-98.

(3) Issa, R.I., 1985, "Solution of the implicitly discretised fluid flow equation by operator splitting", Journal of Computational Physics, Vol. 62, pp. 40-65.

(4) Cooke, C.H., 1986, "An operator splitting for unsteady boundary value problems", Journal of Computational Physics, Vol. 67, pp. 472-478.

(5) Glaister, P., 1988, "An approximate linearised Riemann solver for the three dimensional Euler equations for real gases using operator splitting", Journal of Computational Physics, Vol. 77, pp. 361-383.

(6) Ding, Daoyang, and Liu Philip L.-F, 1989, "An operator splitting algorithm for two-dimensional convection-dispersion-reaction problems", International Journal for Numerical Methods in Engineering, Vol. 28, pp. 1023-1040.

(7) Murphy, J.D., 1985, "Higher order methods for convection-diffusion problems", Computer and Fluids, Vol. 13(2), pp. 157-176.

(8) Mer, Renwei and Plotkin, Allen, 1986, "A finite difference scheme for the solution of the steady Navier-Stokes equations", Computers and Fluids, Vol. 14(3), pp.239-251.

(9) Lai, C.H., Bodvarrson, G.S., and Witherspoon, P.A., 1986, "Second order upwind Differencing Method for non-isothermal chemical transport in porous media", Numerical Heat Transfer, Vol. 9, pp. 453-475.

(10) Liang, Shen-Min and Chau, Jyh-Jany, 1989, "An improved upwind scheme for the Euler equations", Journal of Computational Physics, Vol. 84, pp. 461-473.

(11) Leonard, B.P., and Mokhtari, Simin, 1990, "Beyond first order upwinding: The ULTRA-SHARP alternative for non-oscillatory steady-state simulation of convection", International Journal for Numerical Methods in Engineering, Vol. 30, pp. 729-766.

(12) Raithby, G.D., 1976, "Skew-upstream differencing schemes for problems involving fluid flow", Computer Methods in Applied Mechanics, Vol 9, pp. 151-162.

(13) Patankar, S.V., 1980, Numerical Heat Transfer and Fluid Flow, McGraw Hill Book Company.

(14) Anderson, D.A., Tannehill, T.A. and Pletcher, R.H., 1984, Computational Fluid Mechanics and Heat Transfer, McGraw Hill, New York.

(15) Yanenko, N.N. and Shokin, Yu.I., 1984, Numerical Methods in Fluid Dynamics, MIR Publishers, Moscow.

(16) Euring, R.E., Editor, 1983, Mathematics of Reservoir Simulation, Siam, Philadelphia.

(17) Boberg, Thomas, C., 1988, Thermal Methods of Oil Recovery, An Exxon Monograph, John Wiley and Sons.

(18) Bear, J. and Bachmat, Y., 1990, Introduction to Modeling of Transport Phenomena in Porous Media, Kluwer Academic Publishers.

(19) Odee, A.S., 1969, "Reservoir Simulation. What is it?", Journal of Petroleum Technology, p. 1383.

(20) Spillette, A.G.,1965, "Heat transfer during hot fluid injection into an oil reservoir", Technology, Oct-Dec., Montreal.

(21) Lauweriuer, H.A., 1955, "The transport of heat in an oil layer caused by the injection of hot fluid", Applied Science Research, Sec. A, 5, p. 145.

(22) Weinstein, H.G., Wheeler, J.A., and Woods, E.G., 1977, "Numerical model for steam stimulation", Society of Petroleum Engineering Journal, p. 65.

(23) Shutler, N.D., Dec. 1970, "Numerical 3-Phase Model of the two dimensional steam flood process", Society of Petroleum Engineering Journal, p. 405.

(24) Stevenson, M.D., Kagan, M. and Pinvczewski, 1991, "Computational methods in petroleum reservoir simulation", Computer and Fluid, Vol. 19, No. 1, pp. 19.

(25) Maualem, Yechezkel, 1976, "A new model for predicting the hydraulic conductivity of unsaturated porous media", **Water Resources Research**, Vol. 12, No.3, June, 1976.

(26) Van Genuchten, M.Th., 1980, "A closed form equation for predicting the hydraulic conductivity of unsaturated soils", **Soil Science Society American Journal**, Vol. 44.

(27) Pruess, K. and Narasimhan, T.N., 1982, "On fluid reserves and the production of superheated steam from fractured, vapour-dominated reservoirs", **Journal of Geophysical Research**, Vol. 87, pp. 9329-9339, Nov. 10.

(28) Chavant, G., 1976, "A new formulation of diphasic incompressible flows in porous media", **Lecture Notes in Mathematics 503**, Springer-Verlag, New York.

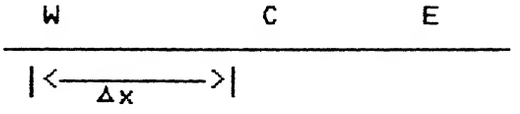
(29) Duff, I.S., 1980, MA 28 - "A set of Fortran subroutines for sparse unsymmetric linear equations", AERE, Harwell, U.K.

APPENDIX A

FALSE DIFFUSION

Consider a one dimensional convection diffusion equation,

$$\frac{\partial \phi}{\partial t} = - \frac{\partial (u\phi)}{\partial x} + \frac{\partial}{\partial x} \left(\Gamma \frac{\partial \phi}{\partial x} \right)$$ where Γ = diffusion coefficient and its solution on grid is shown as:



On discretizing using Central difference scheme for the convection term we get

$$\frac{\partial \bar{\phi}}{\partial t} = \left[-u \frac{\phi_E - \phi_W}{2\Delta x} + \Gamma \left(\frac{\phi_E - \phi_W - 2\phi_C}{\Delta x^2} \right) \right] \quad \text{(i)}$$

where u, Γ and Δx are constant, $\frac{\partial \bar{\phi}}{\partial t}$ is the average value of $\frac{\partial \phi}{\partial t}$ the control volume.

Now we use the upwind scheme for the convective term,

$$\frac{\partial \bar{\phi}}{\partial t} = \left[-u \frac{\phi_C - \phi_L}{\Delta x} + \Gamma \left(\frac{\phi_E - \phi_W - 2\phi_C}{\Delta x^2} \right) \right] \quad \text{(ii)}$$

where

$\phi_L = \phi_w$ if $u > 0$

$= \phi_E$ if $u < 0$

Equation (ii) can be recast into the same form as Equation (i) by writing

$$\frac{\partial \bar{\phi}}{\partial t} = - u \frac{\phi_E - \phi_W}{2\Delta x} + (\Gamma + \Gamma_{num}) \frac{\phi_E + \phi_W - 2\phi_C}{\Delta x^2} \quad \text{(iii)}$$

On comparing (iii) with (i), the additional numerical diffusion coefficient due to upstream differencing is seen to be given by the formula

$$\Gamma_{num} = \frac{|u|\Delta x}{2}$$ and is positive irrespective of the sign of u.

Thus, if one wishes to make Γ_{num} insignificant in comparison with the physical diffusion coefficient Γ, the condition required is $\Gamma_{num}/\Gamma = Pe_g = |u|\Delta x/\Gamma << 2$. This is clearly much more stringent than the practical stability condition $Pe_g \leq 2$ for the central differencing. Both conditions are infact highly unrealistic in terms of practical grid requirements for problems of engineering or geophysical interest, in which the global Peclet or Reynolds number based on a typical macroscopic length scale can be extremely large.

APPENDIX B

(a) Non-dimensionalization of the Heat Transfer equation

The energy equation is of the form,

$$\frac{\partial T}{\partial t} + u\frac{\partial T}{\partial x} + v\frac{\partial T}{\partial T} = \alpha\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right)$$

Let L = characteristic length and U = characteristic velocity.

The characteristic time is L/U.

Let $x^* = x/L$, $t^* = tU/L$, $y^* = y/L$ and

$$T^* = (T - T_{min})/(T_{max} - T_{min})$$

be the nondimensionalized variables.

Then the energy equation in dimensionless variables is,

$$\frac{U}{L}\frac{\partial T^*}{\partial t^*} + \frac{U}{L}u^*\frac{\partial T^*}{\partial x^*} + \frac{U}{L}v^*\frac{\partial T^*}{\partial y^*} = \frac{\alpha}{L^2}(\nabla^{*2} T^*)$$

Hence

$$\left[\frac{\partial T}{\partial t} + u\frac{\partial T}{\partial x} + v\frac{\partial T}{\partial y} = \frac{1}{Pe}\nabla^2 T\right]^* \qquad \text{(i)}$$

where $Pe = \frac{UL}{\alpha}$

The superscript '*' can be dropped in further calculations once it is understood that u,v,T, x and y are dimensionless.

(b) Non dimensionalization of energy equation of Chapter 4

The energy equation used in Chapter 4 has the form,

$$\frac{\partial T}{\partial t} + U\frac{\partial T}{\partial x} + V\frac{\partial T}{\partial y} = \left(\frac{K_h}{\sigma_T}\right)\nabla^2 T$$

where U and V are 'composite' velocities constructed from the individual water and oil velocities. Let $\bar{U}$, L, $\bar{T}$ be the characteristic composite velocity, length and time. $\bar{T} = \frac{L}{\bar{U}}$.

In dimensionless variables, we get

$$\frac{1}{\bar{T}} \frac{\partial T^*}{\partial t^*} + \frac{\bar{U}}{L} U^* \frac{\partial T^*}{\partial x^*} + \frac{\bar{U}}{L} V^* \frac{\partial T^*}{\partial y^*} = \left(\frac{K_h}{\sigma_T}\right) \times \frac{1}{L^2} \nabla^{*2} T^*$$

Hence

$$\left[\frac{\partial T}{\partial t} + U \frac{\partial T}{\partial x} + V \frac{\partial T}{\partial y} = \left(\frac{K_h}{\sigma_T}\right) \frac{1}{L\bar{U}} \nabla^2 T \right]^*$$

where

$$Pe = \frac{\sigma_T L \bar{U}}{K_h}$$

The length scale L cannot be properly identified in this problem. However, simulations in this study are carried out over a region size of 4m and we choose L=1m for the moment.

For a typical case

$$\sigma_T = 2{,}642{,}548.8 \ \frac{W\text{-}sec}{m^3\text{-}^oC} \ , \quad \bar{U} = 1.65 \times 10^{-4} \text{ m/s}$$

and so Pe = 2424.8.

Hence we conclude that Peclet number in the hot water injection method is of the order of 2500 and the flow is convection dominated. As seen in the solutions of the one and two dimensional test problems, this high Pe flow will generate sharp fronts.

APPENDIX C

von-NEUMANN STABILITY ANALYSIS OF THE PREDICTOR STEP IN OS ALGORITHM

The question of weak stability of the predictor would be answered using von Neumann or Fourier analysis. See [4] for details.

The predictor step is, $T_{i,j}^{p+1} = T_{i,j}^{p} - (\frac{u\Delta T}{\Delta x})\,(T_{i,j}^{p} - T_{i-1,j}^{p})$ (i)

Let the error be $e(x,t) = \sum_{m} b_m(t)\, e^{ik_m x}$

Let us study the growth characteristics of a single error term $e_m(x,t) = \left\{ b_m(t)\ e^{ik_m x} \right\}$. (ii)

Now b_m is chosen as e^{at} where $k_m = \frac{m\pi}{L}$ is the spatial wave number.

On substituting (ii) in (i) and cancelling the $e^{at}\ e^{ik_m x}$ term, we get

$$e^{a\Delta t}\ (= G) = (\frac{u\Delta t}{\Delta x}) \left\{ e^{-ik_m \Delta x} - 1 \right\} + 1$$

where $G = \dfrac{e^{a(t+\Delta t)}}{e^{at}}$ is the amplification factor.

Hence $G = (1 - \nu) + \nu\ e^{-i\beta}$ where $\nu = \frac{u\Delta t}{\Delta x}$ and $\beta = k_m \Delta x$

$$|G| = |(1-\nu) + \nu\ \cos\beta - i\nu\sin\beta| = \sqrt{\{1-\nu(1-\cos\beta)\}^2 + \{\nu\sin\beta\}^2}$$

After a little manipulation, this reduces to:

$$|G| = \sqrt{1 + (1 - \cos\beta) \cdot 2\nu \cdot (\nu - 1)}$$

For stability, the amplification factor $|G|$ should be less than 1. Hence $(1 - \cos\beta).2\nu.(\nu-1) < 0$.

Since $\nu > 0$, we require $\nu < 1$.

Hence the condition for stability is

$$0 < \left(\frac{u\Delta t}{\Delta x}\right) < 1$$

APPENDIX D

THE NEED FOR A MACROSCOPIC APPROACH [18]

In principle, the continuum equations that describe various transport pahenomena are known and may be written at the microscopic level within the porous region. Here we focus our attention on what happens at a point within a considered phase such as oil or water present in the porous soil. We may even know the conditions that prevail on the surface that bounds the phase. However, at this level, the equations cannot be solved, since the geometry of the surface that bounds the phase is either not observable or too complex to be described. The same is also true for point values of variables within the phase. As a consequence, the description and solution of a transport problem at the microscopic level is impractical and perhaps impossible.

In many branches of science, it is convenient to ignore the particulate nature of matter and to adopt the hypothesis that matter is a hypothetical substance that is continuous throughout the spatial domain it occupies and can be described in that domain by a set of variables which are continuous and differentiable functions of spatial coordinates and time. This continuum model of matter serves as the fundamental postulate of continuum mechanics and thermodynamics.

The continuum approach, so useful in treating problems related to a single phase, can be extended to a multiphase system such as the porous medium, where the various phases are separated from each other by abrupt interfaces. Accordingly, the

real system, consisting of two, or more phases, that together occupy disjoint subdomains within a porous medium can be replaced by a model in which each of the phases is assumed to behave as a continuum that fills up the entire domain. This domain is termed as the macroscopic space. For each point within this macroscopic space, average values of phase variables are taken over representative elementary volumes (REV), centred at the point regardless of whether, in the real domain this point falls within that phase or outside it. The averaged values are referred to as macroscopic values of the considered variables. By traversing the entire porous medium domain with a moving REV, thus assigning averaged values to every point, we obtain fields of macroscopic variables which are differentiable functions of the space coordinates.

Apart from the advantage of circumventing the need to find exact configuration of the interphase boundaries, the other advantages of the macroscopic approach are as follows.

- It describes processes occurring in porous media in terms of differentiable quantities, thus enabling a solution by employing methods of mathematical analysis.

- The above mentioned quantities are measurable.

These advantages are at the expense of the loss of detailed information concerning microscopic configuration of interphase boundaries and the actual variation of quantities within each phase. But the macroscopic effects of these factors are still retained in the form of coefficients, whose structure and relationship to the statistical properties of the void space

(or phase) configurations can be analyzed and determined. In the case of porous media, the numerical values of these coefficients must be determined experimentally, in the laboratory, or in the field.

Altogether, in view of the advantages of the continuum (or the macroscopic) approach, it shall be employed in Chapter 4 to describe transport phenomena in porous media.

APPENDIX E

DERIVATION OF THE DIFFERENTIAL EQUATIONS GOVERNING OIL-WATER FLOW IN A POROUS MEDIUM

We derive here the equations governing oil and water pressures and temperature in two phase flow through a porous medium

Assumptions:

(i) Two fluid phases flow simultaneously.

(ii) The fluids (oil and water) are immiscible, i.e. there is no mass transfer between the fluids.

(iii) The energy balance assumes instantaneous thermal equilibrium between the fluid and rock matrix. This is justified since the phase velocities are small.

(iv) Gravity term is neglected while calculating Darcy velocities. It is reasonable because the densities of oil and water are close and the time frames are short. As a result, the gravity override effect will not show up in this study.

(v) The surrounding rocks and the matrix are incompressible and have constant transport properties.

(vi) Relative permeabilities of phases is a function of saturation and viscosities are dependent on temperature only.

(vii) There is no net mass source in the flow field.

(viii) Specific heats of the fluids are independent of temperature and fluid pressure.

(ix) Viscous dissipation term in energy equation is neglected.

Derivation:

Following are the basic equations which are used to derive the final pressure and temperature equations for flow and heat transfer in a porous medium. The assumptions given above are used in these derivations. The nomenclature for this Appendix is the same as that of Chapter 4.

Darcy's Law:

$$\tilde{v}_o = - \frac{k\, k_{ro}}{\mu_o} (\nabla \tilde{p}_o) \qquad \text{(i)}$$

$$\tilde{v}_w = - \frac{k\, k_{rw}}{\mu_w} (\nabla \tilde{p}_w) \qquad \text{(ii)}$$

Mass balance:

$$\text{Oil} : \frac{\partial(\varepsilon S_o \rho_o)}{\partial t} + \nabla \cdot \tilde{v}_o \rho_o = q_o \qquad \text{(iii)}$$

$$\text{Water} : \frac{\partial(\varepsilon S_w \rho_w)}{\partial t} + \nabla \cdot \tilde{v}_w \rho_w \qquad q_w \qquad \text{(iv)}$$

Since the rock is incompressible, $\therefore \varepsilon \equiv$ constant. Further q_o and q_w, the mass sources are taken to be zero in this work.

Energy Equation:

Oil and water:

$$\nabla \cdot K_h \nabla T - \nabla \cdot \left[\tilde{v}_o \rho_o H_o + \tilde{v}_w \rho_w H_w \right] = \frac{\partial}{\partial t} \left[\varepsilon (S_o \rho_o H_o + S_w \rho_w H_w \right.$$

$$\left. + (1-\varepsilon)(\rho c)_R T + q_o c_o T_g + q_w c_w T_g \right]$$

Since $H_{o,w} = c_{o,w} T$ and $q_o = q_w \equiv 0$, we get

$$\nabla K_h \nabla T - \nabla \cdot \left[\tilde{v}_o \rho_o c_o T + \tilde{v}_w \rho_w c_w T \right] = \frac{\partial}{\partial t} \left[\varepsilon (S_o \rho_o c_o T + S_w \rho_w c_w T) \right.$$

$$\left. + (1-\varepsilon)(\rho c)_R T \right] \qquad \text{(v)}$$

Equations of state:

$$\xi_w = \frac{1}{\rho_w} \frac{\partial \rho_w}{\partial p_w}\Big|_T \quad , \quad \beta_w = -\frac{1}{\rho_w} \frac{\partial \rho_w}{\partial T}\Big|_{p_w} \qquad \text{(vi)}$$

$$\xi_o = \frac{1}{\rho_o} \frac{\partial \rho_o}{\partial p_o}\Big|_T \quad , \quad \beta_o = -\frac{1}{\rho_o} \frac{\partial \rho_o}{\partial T}\Big|_{p_o}$$

$$p_{c_{ow}}(S_w) = p_o - p_w \qquad \text{(vii)}$$

$$S_o + S_w = 1 \qquad \text{(viii)}$$

We encorporate the equations of state into the mass balance equation for oil. Using (iii) and (vi), we get

$$\varepsilon S_o \frac{\partial \rho_o}{\partial t} + \varepsilon \cdot \rho_o \frac{\partial S_o}{\partial t} + \nabla \cdot \tilde{v}_o \rho_o = 0$$

$$\varepsilon S_o \left\{ \xi_o \rho_o \frac{\partial p_o}{\partial t} - \beta_o \rho_o \frac{\partial T}{\partial t} \right\} + \varepsilon \cdot \rho_o \frac{\partial S_o}{\partial t} + \nabla \cdot \tilde{v}_o \rho_o = 0$$

Using Darcy's law (i) in this equation, we get

$$\epsilon S_o \rho_o \left[\xi_o \frac{\partial p_o}{\partial t} - \beta_o \frac{\partial T}{\partial t}\right] + \epsilon \rho_o \frac{\partial S_o}{\partial t} = \nabla \cdot \frac{k\, k_{ro} \rho_o}{\mu_o} \nabla p_o \qquad \text{(ix)}$$

Similarly for water,

$$\epsilon S_w \rho_w \left[\xi_w \frac{\partial p_w}{\partial t} - \beta_w \frac{\partial T}{\partial t}\right] + \epsilon \cdot \rho_w \frac{\partial S_w}{\partial t} = \nabla \cdot \frac{k\, k_{rw} \rho_w}{\mu_w} \nabla p_w \qquad \text{(x)}$$

Now we would be deriving the two final pressure equations which have to solved simultaneously.

Rewriting (ix) and (x) as

$$-S_o \beta_o \frac{\partial T}{\partial t} + S_o \xi_o \frac{\partial p_o}{\partial t} + \frac{\partial S_o}{\partial t} = \frac{1}{\epsilon \rho_o}\left\{\nabla \cdot \frac{k\, k_{ro} \rho_o}{\mu_o} \nabla p_o\right\}$$

$$-S_w \beta_w \frac{\partial T}{\partial t} + S_w \xi_w \frac{\partial p_w}{\partial t} + \frac{\partial S_w}{\partial t} = \frac{1}{\epsilon \rho_w}\left\{\nabla \cdot \frac{k\, k_{rw} \rho_w}{\mu_w} \nabla p_w\right\}$$

and noting that $p_{c_{ow}}(S_w) = p_o - p_w$,

$$\frac{\partial S_o}{\partial t} = \frac{\partial (1 - S_w)}{\partial t} = \frac{d(1-S_w)}{dp_{c_{ow}}}\left[\frac{\partial p_o}{\partial t} - \frac{\partial p_w}{\partial t}\right],$$

$$= - \frac{d\, S_w}{dp_{c_{ow}}}\left[\frac{\partial p_o}{\partial t} - \frac{\partial p_w}{\partial t}\right],$$

$$\frac{\partial S_w}{\partial t} = \frac{dS_w}{dp_{c_{ow}}}\left[\frac{\partial p_o}{\partial t} - \frac{\partial p_w}{\partial t}\right],$$

we get

$$\text{Oil:}\quad -S_o \beta_o \frac{\partial T}{\partial t} + S_o \xi_o \frac{\partial p_o}{\partial t} - \frac{dS_w}{dp_{c_{ow}}} \left[\frac{\partial p_o}{\partial t} - \frac{\partial p_w}{\partial t}\right]$$

$$= \frac{1}{\rho_o} \nabla \cdot \left[\frac{k\, k_{ro}\, \rho_o}{\mu_o \varepsilon}\right] \nabla p_o \qquad \text{(A)}$$

Similarly,

$$\text{Water:}\quad -S_w \beta_w \frac{\partial T}{\partial t} + S_w \xi_w \frac{\partial p_w}{\partial t} + \frac{dS_w}{dp_{c_{ow}}} \left(\frac{\partial p_o}{\partial t} - \frac{\partial p_w}{\partial t}\right) =$$

$$= \frac{1}{\rho_w} \nabla \cdot \left[\frac{k\, k_{rw}\, \rho_w}{\mu_w \varepsilon}\right] \nabla p_w \qquad \text{(B)}$$

(A) and (B) are the final form of pressure equations. Now, we will transform the energy equation from the conservative form (v) to a nonconservative form using mass balance equations (iii) and (iv). This is necessary to use the OS algoriothm for solving the energy equation. We have,

$$\text{Oil}: \left\{\varepsilon \frac{\partial (S_o \rho_o)}{\partial t} + \nabla \cdot \tilde{v}_o \rho_o = 0\right\} \cdot C_o T$$

$$\text{Water}: \left\{\varepsilon \frac{\partial (S_w \rho_w)}{\partial t} + \nabla \cdot \tilde{v}_w \rho_w = 0\right\} \cdot C_w T$$

On adding these two equations, we get

$$T \frac{\partial}{\partial t} \left\{\varepsilon \left[(1 - S_w)\, c_o \rho_o + c_w S_w \rho_w\right]\right\} + \nabla \cdot \left[\tilde{v}_o \rho_o H_o + \tilde{v}_w \rho_w H_w\right] T$$

$$- (\tilde{v}_o \rho_o c_o \cdot \nabla) T - (\tilde{v}_w \rho_w c_w \cdot \nabla) T = 0 \qquad \text{(xi)}$$

If K_h = constant, then the energy equation (v) can be written as

$$K_h \nabla^2 T - \nabla.\left[\tilde{v}_o \rho_o H_o + \tilde{v}_w \rho_w H_w \right] T = \frac{\partial}{\partial t} \left\{ \varepsilon \left[(1- S_w)\, c_o \rho_o T + c_w \rho_w S_w T \right] + (1 - \varepsilon)(\rho c)_R T \right\} \qquad \text{(xii)}$$

On adding (xi) and (xii), we get

$$K_h \nabla^2 T + T \frac{\partial \left\{ \varepsilon \left[(1-S_w)\, \rho_o\, c_o + \rho_w\, c_w\, S_w \right] \right\}}{\partial t} - \left\{ \left[\tilde{v}_o \rho_o c_o + \tilde{v}_w \rho_w c_w \right] . \nabla \right\} T$$

$$= \frac{\partial}{\partial t} \left\{ \varepsilon \left[(1-S_w) c_o \rho_o T + S_w \rho_w c_w T \right] \right\} + (1-\varepsilon)(\rho c)_R \frac{\partial T}{\partial t}$$

This can be written as,

$$\sigma_T \frac{\partial T}{\partial t} + (\tilde{\sigma} . \nabla)\, T = K_h \nabla^2 T \qquad \text{(xiii)}$$

where

$$\sigma_T = \varepsilon \left\{ (1-S_w) \rho_o c_o + S_w \rho_w c_w \right\} + (1-\varepsilon)\, (\rho c)_R$$

$$\tilde{\sigma} = \tilde{v}_o \rho_o c_o + \tilde{v}_w \rho_w c_w$$

Since $\tilde{v}_o = v_{ox} \hat{i} + v_{oy} \hat{j}$ and $\tilde{v}_w = v_{wx} \hat{i} + v_{wy} \hat{j}$,

therefore

$$\tilde{\sigma} = \sigma_x \hat{i} + \sigma_y \hat{j}$$

where

$$\sigma_x = v_{ox} \rho_o c_o + v_{wx} \rho_w c_w$$

$$\sigma_y = v_{oy} \rho_o c_o + v_{wy} \rho_w c_w$$

In explicit form, Equation (xiii) is,

$$\frac{\partial T}{\partial t} + \left(\frac{\sigma_x}{\sigma_T}\right)\frac{\partial T}{\partial x} + \left(\frac{\sigma_y}{\sigma_T}\right)\frac{\partial T}{\partial y} = \left(\frac{K_h}{\sigma_T}\right)\nabla^2 T$$

This equation is now written in the standard form suitable for the application of the OS algorithm as,

$$\frac{\partial T}{\partial t} + U\frac{\partial T}{\partial x} + V\frac{\partial T}{\partial y} = \left(\frac{K_h}{\sigma_T}\right)\nabla^2 T \qquad (C)$$

where

$$U = \frac{\sigma_x}{\sigma_T} = \frac{v_{ox}\,\rho_o\,c_o + v_{wx}\,\rho_w\,c_w}{\varepsilon\{(1-S_w)\rho_o\,c_o + S_w\,\rho_w\,c_w\} + (1-\varepsilon)(\rho c)_R}$$

and

$$V = \frac{\sigma_y}{\sigma_y} = \frac{v_{oy}\,\rho_o\,c_o + v_{wy}\,\rho_w\,c_w}{\varepsilon\{(1-S_w)\,\rho_o\,c_o + S_w\,\rho_w\,c_w\} + (1-\varepsilon)(\rho c)_R}$$

U and V are the composite velocities of oil and water along x and y directions respectively.

APPENDIX F

(a) We show here that in a transformation from x–y coordinates to $\xi-\eta$ coordinates where ξ is the streamline direction and η is normal to it {See Figure 4.19 (A)},

$$U\,T_x + V\,T_y \equiv U^*\,T_\xi$$

where

$$U^* = \sqrt{U^2 + V^2}$$

At the origin of Figure 4.19(B),

$$\nabla\xi = \xi_x\,\hat{i} + \xi_y\,\hat{j} \quad \text{and} \quad \nabla\eta = \eta_x\,\hat{i} + \eta_y\,\hat{j}$$

$$= \cos\phi\,\hat{i} + \sin\phi\,\hat{j} \qquad = -\sin\phi\,\hat{i} + \cos\phi\,\hat{j}$$

Using chain rule,

$$T_x = T_\xi \cdot \xi_x + T_\eta \cdot \eta_x$$

$$T_y = T_\xi\,\xi_y + T_\eta \cdot \eta_y$$

Substituting the expression in $U\,T_x + V\,T_y$, we get

$$(U\,\xi_x + V\,\xi_y)\,T_\xi + (U\,\eta_x + V\,\eta_y)\,T_\eta$$

$$= (U\cos\phi + V\sin\phi)\,T_\xi + (-U\sin\phi + V\cos\phi)\,T_\eta$$

But $\cos\phi = \dfrac{U}{\sqrt{U^2+V^2}}$, $\sin\phi = \dfrac{V}{\sqrt{U^2+V^2}}$.

Hence,

$$U\,T_x + V\,T_y = \left(\frac{U^2 + V^2}{\sqrt{U^2 + V^2}}\right) T_\xi + \left(\frac{-UV + VU}{\sqrt{U^2 + V^2}}\right) T_\eta$$

$$= \sqrt{U^2 + V^2}\;\; T_\xi$$

$$= U^*\;\; T_\xi\,.$$

(b) Interpolation on a Two Dimensional Grid:

The following method is used to find the temperature at an interior point in a two dimensional grid from the temperatures of the four surrounding nodes. It uses a linear two dimensional interpolating function.

First (x', y') is transformed to (α, β) {See Figure 4.19(E)}

$$\alpha_Q = \frac{2(x_Q - x_1)}{\Delta x} - 1 \; ; \quad -1 \leq \alpha_Q \leq 1$$

$$\beta_Q = \frac{2(y_Q - y_1)}{\Delta y} - 1 \; ; \quad -1 \leq \beta_Q \leq 1$$

Now $T_Q = \sum_{i=1}^{4} N_i T_i$ where N_i is a weighting function.

The N_is are calculated as follows:

$$N_1 = \frac{(1 - \alpha_Q)(1 - \beta_Q)}{4} \; , \; N_2 = \frac{(1 + \alpha_Q)(1 - \beta_Q)}{4}$$

$$N_3 = \frac{(1 + \alpha_Q)(1 + \beta_Q)}{4} \; , \; N_4 = \frac{(1 - \alpha_Q)(1 + \beta_Q)}{4}$$

APPENDIX G

CALCULATING NODAL VELOCITIES FROM NODAL PRESSURES

From Darcy's law (See Section 4.7, Equation 4.6), we get

$$\tilde{v}_o = - \frac{k\, k_{ro}}{\mu_o} \left(\frac{\partial p_o}{\partial x} \hat{i} + \frac{\partial p_o}{\partial y} \hat{j} \right)$$

Hence in component form,

$$v_{ox} = - \theta \frac{\partial p_o}{\partial x}, \quad v_{oy} = - \theta \frac{\partial p_o}{\partial y} \quad \text{where } \theta = \frac{k\, k_{ro}}{\mu_o}$$

Similarly,

$$v_{wx} = - \gamma \frac{\partial p_w}{\partial x}, \quad v_{wy} = - \gamma \frac{\partial p_w}{\partial y} \quad \text{where } \gamma = \frac{k k_{rw}}{\mu_w}$$

Finite difference expression for the pressure gradients at the internal node points ($2 \le i \le N-1$, $2 \le j \le M-1$) is as follows:

$$\frac{\partial p_o}{\partial x} = \frac{p_o\,(i+1,j) - p_o(i-1,j)}{2\Delta x}$$

$$\frac{\partial p_o}{\partial y} = \frac{p_o\,(i,j+1) - p_o(i,j-1)}{2\Delta y}$$

Similarly, expressions for $\frac{\partial p_w}{\partial x}$ and $\frac{\partial p_w}{\partial y}$ can be obtained. At the boundaries, the expressions for pressure gradients are as follows,

$$\left.\frac{\partial p}{\partial x}\right|_{x=0} = \frac{4\,p(2,j) - p(3,j) - 3p(1,j)}{2\Delta x}$$

$$\left.\frac{\partial p}{\partial x}\right|_{x=X} = - \frac{4\,p(N-1,j) - p(N-2,j) - 3\,p(N,j)}{2\Delta x}$$

$$\left.\frac{\partial p}{\partial y}\right|_{y=0} = \frac{4\,p(i,2) - p(i,3) - 3\,p(i,1)}{2\Delta y}$$

$$\left.\frac{\partial p}{\partial y}\right|_{y=Y} = - \frac{4\,p(i,M-1) - p(i,M-2) - 3p(i,M)}{2\Delta y}$$

```
      NAME OF THIS PROGRAM IS NEW.F . IT SOLVES THE 2-D CONVECTION
      DIFFUSION PROBLEM (UNIT VELOCITY ALONG XDIRECTION) USING
      OPERATOR SPLITTING TECHNIQUE . PARABOLIC PART IS SOLVED BY
      A.D.I. ( ALTERNATING DIRECTION IMPLICIT ) TECHNIQUE .
      IT HAS THE FACILITY OF CHANGING THE B.C.S EASILY.

      DIMENSION T1(200,200) , T2(200,200)
      PARAMETER (NODESX=26,NODESY=26)

      DATA T2/40000*0.0/
      NHALF =13
*************************************************************************
      DO 40 I=1,NODESX
      DO 40 J=1,NODESY

      IF(I.EQ.NODESX) T2(I,J)=0.0
      IF(J.EQ.1) T2(I,J)=0.0
      IF(J.EQ.NODESY) T2(I,J)=1.0
      IF(I.EQ.1 .AND. J.GE.NHALF) T2(I,J)=1.0
      IF(I.EQ.1 .AND. J.LT.NHALF) T2(I,J)=0.0

)     CONTINUE
*************************************************************************

      DO 50 I=1,NODESX
      DO 50 J=1,NODESY
0     T1(I,J) = T2(I,J)

      XMAX=2.0
      YMAX=2.0
      PRINT * , 'WHAT IS PE , DELT?'
      READ * , PE,DELT
      DELX = XMAX /(NODESX-1)
      DELY = YMAX /(NODESY-1)
      TIME=0.5
      DELT1 = DELT
      DELT2 = DELT
      NTIMESTEPS = IFIX (TIME/DELT)

      CALL PRINTDATA (PE,TIME,DELT,DELX,DELY,XMAX,YMAX,NODESX,
   1                  NODESY,NTIMESTEPS)

      DO I=1,NTIMESTEPS
       CALL STEP1(T1,T2,NODESX,NODESY,DELT1,DELX)
       CALL STEP2 (T1,T2,DELX,DELY,DELT2,PE,NODESX,NODESY,NHALF)
      END DO

      CALL PRINTOUTPUT(T2,NODESX,NODESY)

      END

      SUBROUTINE STEP1 (T1,T2,NX,NY,DELT,DELX)
      DIMENSION T1(200,200),T2(200,200)

      R=DELT/DELX

      DO I = 2 , NX-1
          DO J = 2 , NY-1
            T1(I,J) = R*T2(I-1,J)+(1.0-R)*T2(I,J)
          END DO
      END DO

      RETURN
      END

      SUBROUTINE STEP2 (T1,T2,DELX,DELY,DELT2,PE,NODESX,NODESY,NHALF)
```

```
      DIMENSION T1(200,200),T2(200,200)
      REAL INTERTEMP(200,200),AR(200),BR(200),CR(200),FR(200),TEMP3(200)
      DELTADI = DELT2 / 2.0
      TEMP1 = DELTADI / (PE*DELX*DELX)
      TEMP2 = DELTADI / (PE*DELY*DELY)
      BLEFT = 1 + 2*TEMP1
      BRIGHT = -(2*TEMP2)
      A = -TEMP1
      C = -TEMP1
      D = -TEMP2
      E = -TEMP2

      DO I=1,200
        AR(I) = A
        BR(I) = BLEFT
        CR(I) = C
      END DO
      DO J = 2, NODESY-1
           DO I= 2 , NODESX-1
           FR(I-1)=(1+BRIGHT)*T1(I,J)-D*T1(I,J+1)-E*T1(I,J-1)
           END DO
           FR(1) = FR(1) - A*T1(1,J)
           FR(NODESX-2) = FR(NODESX-2) - C*T1(NODESX,J)
           CALL TRIDIA (AR,BR,CR,FR,(NODESX-2),TEMP3)
           DO I=2, NODESX
              INTERTEMP(I,J) = TEMP3(I-1)
           END DO
      END DO
*****************************************************************************
      DO 20 I=1,NODESX
      DO 20 J=1,NODESY

      IF(I.EQ.NODESX) INTERTEMP(I,J)= 0.0
      IF(J.EQ.1) INTERTEMP(I,J)=0.0
      IF(J.EQ.NODESY) INTERTEMP(I,J)=1.0
      IF(I.EQ.1 .AND. J.GE.NHALF) INTERTEMP(I,J)=1.0
      IF(I.EQ.1 .AND. J.LT.NHALF) INTERTEMP(I,J)=0.0

20    CONTINUE
*****************************************************************************

      DO 30 I=1,NODESX
      DO 30 J=1,NODESY
30    T2(I,J) = INTERTEMP(I,J)

      BLEFT = 1 + 2*TEMP2
      BRIGHT = -2*TEMP1
      DO I=1,200
         AR(I) = E
         BR(I) = BLEFT
         CR(I) = D
      END DO
      DO I = 2 , NODESX-1
           DO J=2, NODESY-1
           FR(J-1)=(1+BRIGHT)*INTERTEMP(I,J)-C*INTERTEMP(I+1,J)-
     *                A*INTERTEMP(I-1,J)
           END DO
           FR(1) = FR(1) - E*INTERTEMP(I,1)
           FR(NODESY-2) = FR(NODESY-2) - D*INTERTEMP(I,NODESY)
           CALL TRIDIA (AR,BR,CR,FR,(NODESY-2),TEMP3)
           DO J=2,NODESY-1
              T2(I,J) = TEMP3(J-1)
           END DO
      END DO

      RETURN
      END
```

```
      implicit real (k)
      real infinity
      parameter (n=41,m=5,l=2*n*m,biglim=0.1,itrnslim=20)
      parameter (infinity=987654321.0,delta=0.000001)
      parameter (nd=17) ! Domain dimensions.
      dimension po1(50,10),pw1(50,10)
      dimension po2(50,10),pw2(50,10),po2old(50,10)
      dimension pw2old(50,10),sw(50,10),vdummy(50,10)
      dimension vox(50,10),voy(50,10),vwx(50,10),vwy(50,10)
      dimension U(50,10),V(50,10)
      dimension told(50,10),t1(50,10),tm(50,10)
      dimension t2(50,10),t2old(50,10),sigma_t(50,10)
      dimension denoil(50,10),denwater(50,10)
      dimension A(61000),r(450),wm(450)
      integer*4 ikeep(2100),iw(3300),ivect(61000),jvect(61000)
      integer*4 irn(61000),icn(122000),iglb(450,450)

      common /area1/delx,dely,delt1,Bi,ti
      common /area2/tbc_l,tbc_r
      common /area3/t1,tm,t2,sigma_t,Kh
  Conversion factors.
      conv_psi = 6.894757e+3    ! Conversion factor(psi to pascal)
      conv_fah = 1.0/1.8        ! Conversion factor(fahrenheit to celsius)
      conv_mpa = 1.0e+6         ! Conversion factor(megapascal to pascal)
      conv_darcy = 9.87e-13     ! Conversion factor(darcy to meter-squared)
      conv_hour = 3600          ! Conversion factor(hour to sec)
  Parameter values . Beta values are to be checked.
      zeta_oil = 5.0e-6*(1/conv_psi)    ! unit = 1/pascal
      zeta_water = 3.1e-6*(1/conv_psi) ! unit = 1/pascal
      beta_oil = 0.00041*(1/conv_fah)  ! unit = 1/celsius
      beta_water = 0.00041*(1/conv_fah)! unit = 1/celsius
      epsilon = 0.375                  ! unit = dimensionless
      k = 132.0*conv_darcy             ! unit = m**2

*****************************************************************
        pw = 190.0 * conv_psi / conv_mpa          ! unit = mpa
        tbc_l = 100.0                             ! unit = celsius
        tbc_r = 50.0                              ! unit = celsius
        data t1,t2,denoil,denwater/500*50.0,500*50.0,500*900.0,
     1  500*1000.0/
      do i=1,n
      do j=1,m
        pw1(i,j) = pw                             ! unit = mpa
        sw(i,j)  = 0.2                            ! unit = dimensionless
        call capillarypressure (sw(i,j),pc,ierror)
        po1(i,j) = (pc/conv_mpa) + pw1(i,j)       ! unit = mpa

        pw1(1,j) = 260.0 * conv_psi / conv_mpa    ! unit = mpa
        sw(1,j)  = 0.86                           ! unit = dimensionless
        call capillarypressure (sw(1,j),pc,ierror)
        po1(1,j) = (pc/conv_mpa) + pw1(1,j)       ! unit = mpa

        pw1(2,j) =  242.5 * conv_psi / conv_mpa   ! unit = mpa
        sw(2,j)  = .695
        call capillarypressure (sw(2,j),pc,ierror)
        po1(2,j) = (pc/conv_mpa) + pw1(2,j)       ! unit = mpa

        pw1(3,j) = 225.0 * conv_psi / conv_mpa    ! unit = mpa
        sw(3,j)  = .53
        call capillarypressure (sw(3,j),pc,ierror)
        po1(3,j) = (pc/conv_mpa) + pw1(3,j)       ! unit = mpa

        pw1(4,j) = 207.5 * conv_psi / conv_mpa    ! unit = mpa
        sw(4,j)  = .365
        call capillarypressure (sw(4,j),pc,ierror)
        po1(4,j) = (pc/conv_mpa) + pw1(4,j)       ! unit = mpa
```

```
        t1(i,j) = tbc_r                                  ! unit = celsius
        t1(1,j) = tbc_l                                  ! unit = celsius
        t1(2,j) = (tbc_l-tbc_r)*3.0/4.0 + tbc_r
        t1(3,j) = (tbc_l+tbc_r) / 2.0                    ! unit = celsius
        t1(4,j) = (tbc_l-tbc_r)*1.0/4.0 + tbc_r
      end do
      end do
      do i=1,n
      do j=1,m
        call oildensity (po1(i,j)*conv_mpa ,t1(i,j),deno)
        call waterdensity (pw1(i,j)*conv_mpa ,t1(i,j),denw)
        denoil(i,j) = deno                               ! unit = kg/cubic_meter
        denwater(i,j) = denw                             ! unit = kg/cubic_meter
      end do
      end do

      delt = 0.01*conv_hour                              ! unit = sec
      time = 1.0*conv_hour                               ! unit = sec
      delx = 0.1                                         ! unit = meter
      dely = 0.2                                         ! unit = meter
      Bi = 0.01
      ti = tbc_r                                         ! unit = celsius
*****************************************************************
      sum = 0.0
      do 82 i=1,nd
      do 82 j=1,m
      so = 1.0-sw(i,j)
2     sum = sum+so
      so_avgI = sum / float(nd*m)

      ntimesteps = ifix(time/delt)
      xmax = (n-1)*delx
      ymax = (m-1)*dely
      call printdata(time,delt,delx,dely,xmax,ymax,n,m,
     1                ntimesteps)
      write(*,1115)nd,m,biglim,itrnslim,Bi,tbc_l,tbc_r,ti
115   format('    DOMAIN NODES (X-DIRECTION) =',I3/
     1       '    DOMAIN NODES (Y-DIRECTION) =',I3/
     2       '                        BIGLIM =',F4.2/
     3       '                     ITRNSLIM =',I3/
     4       '                  BIOT NO.(Bi) =',F6.3/
     5       '       INLET WATER TEMPERATURE =',F5.1/
     6       'INITIAL FORMATION TEMPERATURE =',F5.1/
     7       '                           Ti =',F5.1/)

      nh= 2*m

c  Giving values to IGLB.
      nz=0
      do 10 j=1,nh+1
      do 10 i=1,l-j+1          ! lower
      nz=nz+1
      irn(nz)=i+j-1
      icn(nz)=i
      IGLB(irn(nz),icn(nz)) = nz
10    continue
      do 20 j=2,nh+1
      do 20 i=1,l-j+1
      nz=nz+1                  ! upper
      irn(nz)=i
      icn(nz)=i+j-1
      IGLB(irn(nz),icn(nz)) = nz
20    continue
```

```
      licn = 2*nz + 5
      lirn = nz + 5
      mtype = 1
      uf = 0.1
      do 68 i=1,nz
      ivect(i)=irn(i)
      jvect(i)=icn(i)
      continue

      iter = 0
      V_wsum = 0.0
      V_osum = 0.0
        do 72 i=1,n
        do 72 j=1,m
        told(i,j) = t1(i,j)
        po2old(i,j) = po1(i,j)
        pw2old(i,j) = pw1(i,j)
        t2old(i,j) = t1(i,j)
        po2(i,j) = po1(i,j)
        pw2(i,j) = pw1(i,j)
        t2(i,j) = t1(i,j)
2
       do 2000 nt=1,ntimesteps

       delt1=delt
       nsubtimesteps = 1
       if(nt .eq. 1) then
         delt1=delt/10.0
         nsubtimesteps = 10
       endif
       if(nt .eq. 2) then
         delt1=delt/5.0
         nsubtimesteps = 5
       endif
       if(nt .eq. 3) then
         delt1=delt/2.0
         nsubtimesteps = 2
       endif

       do 1000 nt1=1,nsubtimesteps

       bigmax = 0.0
       iterold = iter

500    continue

       do 71 i=1,n
       do 71 j=1,m
       po2old(i,j) = po2(i,j)
       pw2old(i,j) = pw2(i,j)
71     t2old(i,j) = t2(i,j)
c  Initialising.
       do 15 i=1,nz
       A(i)=0.0
       if(i.gt.1) go to 15
       R(i)=0.0
15     continue
c  Boundary Conditions.
       do 110 iflag=1,4
       if(iflag.eq.1) then
         ilower= 1                    ! left b.c.
         iupper= 2*m
         istep=2
         go to 109
       endif
       if(iflag.eq.2) then
```

```
          ilower= 2*(n-1)*m+1    ! right b.c.
          iupper= 2*m*n
          istep = 2
          go to 109
        endif
        if(iflag.eq.3) then
          ilower= 2*m+1
          iupper= 2*(n-2)*m+1    ! bottom b.c.
          istep = 2*m
          go to 109
        endif
        if(iflag.eq.4) then
          ilower= 2*2*m-1
          iupper= (n-1)*2*m-1    ! top b.c.
          istep = 2*m
          go to 109
        endif

109     do 110 i=ilower,iupper,istep
        irem =mod(i,2)
        if(irem.ne.0) then
          in = (i+1)/2
        else
          in = i/2
        endif
        iy = mod(in,m)
        if(iy.eq.0) iy = m
        ix = in/m + 1
        if(mod(in,m).eq.0) ix = in/m
        r(i)= po1(ix,iy)
        r(i+1)= pw1(ix,iy)
        j1=i-nh
        j2=i+nh
        if(j1.lt.1) j1=1
        if(j2.gt.1) j2=1
        do 111 j=j1,j2
        A(IGLB(i,j)) = 0.0
        if(i.eq.j) A(IGLB(i,j)) = 1.0
111     continue
        if(j1.eq.1) then
           j2=j2+1
           go to 112
        endif
        if(j2.eq.1) then
           j1=j1+1
           go to 112
        endif
        j1=j1+1
        j2=j2+1
112     do 110 j=j1,j2
        A(IGLB(i+1,j)) = 0.0
        if(i+1.eq.j) A(IGLB(i+1,j)) = 1.0
110     continue

        do 200 i=2,n-1
        do 200 j=1,m

c       COMPUTING Ao,Bo,Co,Do,Eo,Fo AND Go VALUES

        call oilrelpermeability (sw(i,j),kr1,ierror)
        call oilviscosity (t2(i,j),viscoil1,ierror)
        if(kr1 .gt. delta) then
          term1 = viscoil1 / (kr1*denoil(i,j))
        else
          term1 = infinity
        endif
```

```
c  Computing the values of theta towards the east of the node.
        call oilrelpermeability(sw(i+1,j),kr2,ierror)
        call oilviscosity (t2(i+1,j),viscoil2,ierror)
        if(kr2 .gt. delta) then
          term2 = viscoil2 / (kr2*denoil(i+1,j))
        else
          term2 = infinity
        endif
        if(term1.lt.(infinity/2.0).and.term2.lt.(infinity/2.0)) then
          term = (term1+term2) / 2.0
          temp = 1.0 / term
        else
          temp = 0.0
        endif
        theta_e = (k/epsilon)*(temp)
c  Computing the values of theta towards the west of the node.
        call oilrelpermeability(sw(i-1,j),kr2,ierror)
        call oilviscosity (t2(i-1,j),viscoil2,ierror)
        if(kr2 .gt. delta) then
          term2 = viscoil2 / (kr2*denoil(i-1,j))
        else
          term2 = infinity
        endif
        if(term1.lt.(infinity/2.0).and.term2.lt.(infinity/2.0)) then
          term = (term1+term2) / 2.0
          temp = 1.0 / term
        else
          temp = 0.0
        endif
        theta_w = (k/epsilon)*temp
c  Computing the values of theta towards the north of the node.
        if(j.eq.m) go to 401
        call oilrelpermeability(sw(i,j+1),kr2,ierror)
        call oilviscosity (t2(i,j+1),viscoil2,ierror)
        if(kr2 .gt. delta) then
          term2 = viscoil2 / (kr2*denoil(i,j+1))
        else
          term2 = infinity
        endif
        if(term1.lt.(infinity/2.0).and.term2.lt.(infinity/2.0)) ther
          term = (term1+term2) / 2.0
          temp = 1.0 / term
        else
          temp = 0.0
        endif
        theta_n = (k/epsilon)*temp
401     continue
c  Computing the values of theta towards the south of the node.
        if(j.eq.1) go to 402
        call oilrelpermeability(sw(i,j-1),kr2,ierror)
        call oilviscosity (t2(i,j-1),viscoil2,ierror)
        if(kr2 .gt. delta) then
          term2 = viscoil2 / (kr2*denoil(i,j-1))
        else
          term2 = infinity
        endif
        if(term1.lt.(infinity/2.0) and.term2.lt.(infinity/2.0)) then
          term = (term1+term2) / 2.0
          temp = 1.0 / term
        else
          temp = 0.0
        endif
        theta_s = (k/epsilon)*temp
402     continue
        if (j.eq.1)  theta_s = theta_n
        if (j.eq.m)  theta_n = theta_s
```

```
c  Computing the Ao values.
       anum = -theta_e
       denom = denoil(i,j)*delx*delx
       Ao = anum/denom
c  Computing the Bo values.
       so = 1.0-sw(i,j)
       pc = po2(i,j)-pw2(i,j)
       call sw_gradient(pc*conv_mpa,grad,ierror)
       temp1 = (so*zeta_oil-grad)/delt1
       anum = theta_e+theta_w
       denom = denoil(i,j)*delx*delx
       temp2 = anum/denom
       anum = theta_n+theta_s
       denom = denoil(i,j)*dely*dely
       temp3 = anum/denom
       Bo = temp1+temp2+temp3
c  Computing the Co values.
       Co = grad/delt1
c  Computing the Do values.
       Do = -theta_w/(denoil(i,j)*delx*delx)
c  Computing the Eo values.
       Eo = -theta_n/(denoil(i,j)*dely*dely)
c  Computing the Fo values.
       Fo = -theta_s/(denoil(i,j)*dely*dely)
c  Computing the Go values.
       temp2 = so*beta_oil*(t1(i,j)-told(i,j)) / delt1
       temp3 = (temp1*po1(i,j)+Co*pw1(i,j)) * conv_mpa
       Go = temp2 + temp3

c      COMPUTING Aw,Bw,Cw,Dw,Ew,Fw AND Gw VALUES.

       call waterrelpermeability(sw(i,j),kr1,ierror)
       call waterviscosity (t2(i,j),viscwater1,ierror)
       if(kr1 .gt. delta) then
         term1 = viscwater1 / (kr1*denwater(i,j))
       else
         term1 = infinity
       endif
c  Computing the values of gamma towards the east of the node.
       call waterrelpermeability(sw(i+1,j),kr2,ierror)
       call waterviscosity (t2(i+1,j),viscwater2,ierror)
       if(kr2 .gt. delta) then
         term2 = viscwater2 / (kr2*denwater(i+1,j))
       else
         term2 = infinity
       endif
       if(term1.lt.(infinity/2.0).and.term2.lt.(infinity/2.0)) then
         term = (term1+term2) / 2.)
         temp = 1.0 / term
       else
         temp = 0.0
       endif
       gamma_e = (k/epsilon)*(temp)
c  Computing the values of gamma towards the west of the node.
       call waterrelpermeability(sw(i-1,j),kr2,ierror)
       call waterviscosity (t2(i-1,j),viscwater2,ierror)
       if(kr2 .gt. delta) then
         term2 = viscwater2 / (kr2*denwater(i-1,j))
       else
         term2 = infinity
       endif
       if(term1.lt.(infinity/2.0).and.term2.lt.(infinity/2.0)) then
         term = (term1+term2) / 2.0
         temp = 1.0 / term
       else
```

```
          temp = 0.0
        endif
        gamma_w = (k/epsilon)*temp
c  Computing the values of gamma towards the north of the node.
        if(j.eq.m) go to 403
        call waterrelpermeability(sw(i,j+1),kr2,ierror)
        call waterviscosity (t2(i,j+1),viscwater2,ierror)
        if(kr2 .gt. delta) then
          term2 = viscwater2 / (kr2*denwater(i,j+1))
        else
          term2 = infinity
        endif
        if(term1.lt.(infinity/2.0).and.term2.lt.(infinity/2.0)) then
          term = (term1+term2) / 2.0
          temp = 1.0 / term
        else
          temp = 0.0
        endif
        gamma_n = (k/epsilon)*temp
403     continue
c  Computing the values of gamma towards the south of the node.
        if(j.eq.1) go to 404
        call waterrelpermeability(sw(i,j-1),kr2,ierror)
        call waterviscosity (t2(i,j-1),viscwater2,ierror)
        if(kr2 .gt. delta) then
          term2 = viscwater2 / (kr2*denwater(i,j-1))
        else
          term2 = infinity
        endif
        if(term1.lt.(infinity/2.0).and.term2.lt.(infinity/2.0)) then
          term = (term1+term2) / 2.0
          temp = 1.0 / term
        else
          temp = 0.0
        endif
        gamma_s = (k/epsilon)*temp
404     continue
        if (j.eq.1) gamma_s = gamma_n
        if (j.eq.m) gamma_n = gamma_s

c  Computing the Aw values.
        anum = -gamma_e
        denom = denwater(i,j)*delx*delx
        Aw = anum/denom
c  Computing the Bw values.
        temp1 = (sw(i,j)*zeta_water-grad)/delt1
        anum = gamma_e+gamma_w
        denom = denwater(i,j)*delx*delx
        temp2 = anum/denom
        anum = gamma_n+gamma_s
        denom = denwater(i,j)*dely*dely
        temp3 = anum/denom
        Bw = temp1+temp2+temp3
c  Computing the Cw values.
        Cw = grad/delt1
c  Computing the Dw values
        Dw = -gamma_w/(denwater(i,j)*delx*delx)
c  Computing the Ew values
        Ew = -gamma_n/(denwater(i,j)*dely*dely)
c  Computing the Fw values.
        Fw = -gamma_s/(denwater(i,j)*dely*dely)
c  Computing the Gw values.
        temp2 = sw(i,j)*beta_water*(t1(i,j) - told(i,j)) / delt1
        temp3 = (temp1*pw1(i,j)+Cw*po1(i,j)) * conv_mpa
        Gw = temp2 + temp3
```

```
      Ao = Ao*conv_mpa
      Bo = Bo*conv_mpa
      Co = Co*conv_mpa
      Do = Do*conv_mpa
      Eo = Eo*conv_mpa
      Fo = Fo*conv_mpa

      Aw = Aw*conv_mpa
      Bw = Bw*conv_mpa
      Cw = Cw*conv_mpa
      Dw = Dw*conv_mpa
      Ew = Ew*conv_mpa
      Fw = Fw*conv_mpa

      if(j.eq.1) then
      Ao = 0.0
      Bo = 1.0
      Co = 0.0
      Do = 0.0
      Eo = -1.0
      Fo = 0.0
      Go = 0.0

      Aw = 0.0
      Bw = 1.0
      Cw = 0.0
      Dw = 0.0
      Ew = -1.0
      Fw = 0.0
      Gw = 0.0
      endif

      if(j.eq.m) then
      Ao = 0.0
      Bo = 1.0
      Co = 0.0
      Do = 0.0
      Eo = 0.0
      Fo = -1.0
      Go = 0.0

      Aw = 0.0
      Bw = 1.0
      Cw = 0.0
      Dw = 0.0
      Ew = 0.0
      Fw = -1.0
      Gw = 0.0
      endif

c  Assigning Ao,Bo,Co,Do,Eo,Fo,Go and Aw,Bw,Cw,Dw,Ew,Fw,Gw values
c  to the respective A(i,j) slots and r(i).

      nz1 = j+(i-1)*m     !Keeps track of the node in the grid.
      nza = 2*nz1         !Keeps track of the slot in A(i,j).
      A(IGLB(nza-1,nza-1))      = Bo
      A(IGLB(nza-1,nza))        = Co
      A(IGLB(nza-1,nza+1))      = Eo
      A(IGLB(nza-1,nza-1+2*m)) = Ao
      A(IGLB(nza-1,nza-1-2*m)) = Do
      A(IGLB(nza-1,nza-3))      = Fo
      r(nza-1)                  = Go

      A(IGLB(nza,nza))        = Bw
      A(IGLB(nza,nza-1))      = Cw
      A(IGLB(nza,nza-2))      = Fw
```

```
        A(IGLB(nza,nza-2*m))    = Dw
        A(IGLB(nza,nza+2))      = Ew
        A(IGLB(nza,nza+2*m))    = Aw
        r(nza)                  = Cw

200     continue

        if(iter.gt.0) go to 55
        call ma28a(l,nz,A,licn,irn,lirn,icn,uf,ikeep,iw,wm,iflaga)
        if(iflaga.ne.0) print *,'iflaga=',iflaga

        if(iter.eq.0) go to 56
55      continue
        call ma28b(l,nz,a,licn,ivect,jvect,icn,ikeep,iw,wm,iflagb)
        if(iflagb.ne.0) print *,'iflagb =',iflagb

56      continue
        call ma28c(l,a,licn,icn,ikeep,r,wm,mtype)

c  Assigning solution r(i) to po2(i,j) and pw2(i,j).
        do 57 i=1,l
        irem =mod(i,2)
        if(irem.ne.0) then
          in = (i+1)/2
          iflag = 1
        else
          in = i/2
          iflag = 2
        endif
        iy = mod(in,m)
        if(iy.eq.0) iy = m
        ix = in/m + 1
        if(mod(in,m).eq.0) ix = in/m
        if(iflag.eq.1) po2(ix,iy) = r(i)
        if(iflag.eq.2) pw2(ix,iy) = r(i)
57      continue

c  Updating sw(i,j) , denoil(i,j) and denwater(i,j)
c  (due to the pressure).

        do 58 i=2,n-1
        do 58 j=1,m
        p = po2(i,j) * conv_mpa
        t = t2(i,j)
        call oildensity (p,t,densitynew)
        denoil(i,j) = densitynew

        p = pw2(i,j) * conv_mpa
        call waterdensity (p,t,densitynew)
        denwater(i,j) = densitynew

        pc = po2(i,j)-pw2(i,j)
        call watersaturation (pc*conv_mpa,sw(i,j),ierror)
58      continue

c  Determination of Darcy velocities.
        do 59 i=1,n
        do 59 j=1,m
        call oilviscosity (t2(i,j),viscoil,ierror)
        call oilrelpermeability(sw(i,j),kro,ierror)
        theta = (k*kro)/viscoil
        call waterviscosity (t2(i,j),viscwater,ierror)
        call waterrelpermeability(sw(i,j),krw,ierror)
        gamma = (k*krw)/viscwater

        if(i.eq.1 .or  i.eq.n .or. j.eq.1 .or  j.eq.m) then
```

```
if((i.eq.1 .or. i.eq.n) .and. (j.ne.1 .and. j.ne.m)) then
  if(i.eq.1) index=1
  if(i.eq.n) index=2
  call edgegrad (pograd,po2,i,j,n,m,index)
  call edgegrad (pwgrad,pw2,i,j,n,m,index)
  otempx = conv_mpa * pograd
  wtempx = conv_mpa * pwgrad
  otempy = ((po2(i,j+1)-po2(i,j-1)) * conv_mpa)/ (2*dely)
  wtempy = ((pw2(i,j+1)-pw2(i,j-1)) * conv_mpa)/ (2*dely)
  go to 61
endif
if((j.eq.1 .or. j.eq.m) .and. (i.ne.1 .and. i.ne.n)) then
  if(j.eq.1) index = 3
  if(j.eq.m) index = 4
  call edgegrad (pograd,po2,i,j,n,m,index)
  call edgegrad (pwgrad,pw2,i,j,n,m,index)
  otempx = ((po2(i+1,j)-po2(i-1,j)) * conv_mpa)/ (2*delx)
  wtempx = ((pw2(i+1,j)-pw2(i-1,j)) * conv_mpa)/ (2*delx)
  otempy = conv_mpa * pograd
  wtempy = conv_mpa * pwgrad
  go to 61
endif
if(i.eq.1) index = 1
if(i.eq.n) index = 2
call edgegrad (pograd,po2,i,j,n,m,index)
call edgegrad (pwgrad,pw2,i,j,n,m,index)
otempx = conv_mpa * pograd
wtempx = conv_mpa * pwgrad
if(j.eq.1) index = 3
if(j.eq.m) index = 4
call edgegrad (pograd,po2,i,j,n,m,index)
call edgegrad (pwgrad,pw2,i,j,n,m,index)
otempy = conv_mpa * pograd
wtempy = conv_mpa * pwgrad
go to 61
endif
otempx = ((po2(i+1,j)-po2(i-1,j)) * conv_mpa)/ (2*delx)
wtempx = ((pw2(i+1,j)-pw2(i-1,j)) * conv_mpa)/ (2*delx)
otempy = ((po2(i,j+1)-po2(i,j-1)) * conv_mpa)/ (2*dely)
wtempy = ((pw2(i,j+1)-pw2(i,j-1)) * conv_mpa)/ (2*dely)

continue

vox(i,j) = - theta*otempx
voy(i,j) = - theta*otempy
vwx(i,j) = - gamma*wtempx
vwy(i,j) = - gamma*wtempy

continue

SOLUTION OF THE ENERGY EQUATION.

co =0.5*4184.0       ! specific heat of the oil (W-sec/Kg---degC)
cw = 4184.0          ! specific heat of water (W-sec/Kg---degC)
Hr = 2412996.5       ! heat-capacity / volume (W-sec/m**3-degC)
                     ! of the porous medium.
Kh = 40.0*4.1528e-3  ! Thermal conductivity of the porous
                     ! medium (W/m-degC)

do 60 i=1,n
do 60 j=1,m
so = 1-sw(i,j)
temp1 = epsilon*( so*co*denoil(i,j) + sw(i,j)*cw*denwater(i,j) )
temp2 = (1-epsilon)*Hr
sigma_t(i,j) = temp1 + temp2
sigma_x = vox(i,j)*denoil(i,j)*co + vwx(i,j)*denwater(i,j)*cw
```

```
      sigma_y = voy(i,j)*denoil(i,j)*co + vwy(i,j)*denwater(i , j)*cw
c U and V are the pseudovelocities.
      U(i,j) = sigma_x / sigma_t(i,j)
      V(i,j) = sigma_y / sigma_t(i,j)
c PREDICRTOR : Temperature_transfer is a subroutine that transfers the last
c timestep temperature from the point q,which is upstream of the point p,
c to the point p . This is the solution of the hyperbolic part .

      if(i.ge.2 .and. i.le.n-1 .and. j.ge.2 .and. j.le.m-1) then
        call step1 (U(i,j),V(i,j),i,j,tm(i,j))
      else
        if((j.eq.1 .or. j.eq.m).and. i.ne.1 .and. i.ne.n ) then
          call step1b (U(i,j),i,j,tm(i,j))
        else
          tm(i,j) = t2(i,j)
        endif
      endif
60    continue

c CORRECTOR : This subsequent part will solve the parabolic part
c implicitly using A.D.I. and T.D.M.A.

        call step2 (n,m)

c Updating oil and water densities (due to the temperature).
      do 76 i=1,n
      do 76 j=1,m
      p = po2(i,j) * conv_mpa
      t = t2(i,j)
      call oildensity (p,t,densitynew)
      denoil(i,j) = densitynew

      p = pw2(i,j) * conv_mpa
      call waterdensity (p,t,densitynew)
      denwater(i,j) = densitynew
76    continue

      iter = iter + 1
      itrns = iter-iterold
      big = 0.0
      do 77 i=1,n
      do 77 j=1,m
      perror1 = 100.0* abs((po2(i,j)-po2old(i,j)) / po2(i,j))
      perror2 = 100.0*abs((pw2(i,j)-pw2old(i,j)) / pw2(i,j))
      perror3 = 100.0*abs((t2(i,j)-t2old(i,j)) / t2(i,j))
      big = amax1(big,perror1,perror2,perror3)
77    continue
      bigmax = amax1(big,bigmax)
      if(big.gt.biglim .and. itrns.le.itrnslim) go to 500

      do 40 i=1,n
      do 40 j=1,m
      told(i,j) = t1(i,j)
      po1(i,j) = po2(i,j)
      pw1(i,j) = pw2(i,j)
40    t1(i,j) = t2(i,j)

1000  continue

c Computation of percentage pore-volume (ppv) of oil left.
      sum = 0.0
      do 62 i=1,nd
      do 62 j=1,m
      so = 1.0-sw(i,j)
62    sum = sum+so
      so_avg = sum / float(nd*m)
```

```
        ppv = (so_avgI - so_avg) * 100
c  Computation of time-averaged centre-line  velocities for the oil and
c  the water phases. Oil velocity is calculated at x=L and water velocity
c  is calculated at x=L/2 (L = domain_length)
        V_wsum = V_wsum + vwx(nd/2,(m+1)/2)
        V_osum = V_osum + vox(nd,(m+1)/2)
        V_w = V_wsum / float(nt)
        V_o = V_osum / float(nt)

        isee = mod(nt,5)
        if(isee .eq. 0) then
        print *,'timestep =',nt
        print *,'sub-timestep =',nt1
        print *,'itrns',itrns
        print *,'big',big
        print *,'sw'
        do 107 j=(m+1)/2,(m+1)/2
107     write (*,1111) (sw(i,j),i=1,n)
1111    format(10(2x,f6.4))
        print *,'vox'
        supfactor = 1e-5
        do 108 j=(m+1)/2,(m+1)/2
108     write (*,1112) ((vox(i,j)/supfactor),i=1,nd,2)
1112    format(10(2x,f6.3))
        print *,'vwx'
        supfactor = 1e-4
        do 106 j=(m+1)/2,(m+1)/2
106     write (*,1113) ((vwx(i,j)/supfactor),i=1,nd,2)
1113    format(10(2x,f6.3))
        print *,'t'
        do 113 j=(m+1)/2,(m+1)/2
113     write (*,1114) (t1(i,j),i=1,nd)
1114    format(10(1x,f7.3))
        print *,'ppv',ppv
        print *,'   '
        endif
2000    continue
        end

        subroutine step1 (U,V,i,j,tp)
        parameter (pi=3.1415)
        common /area1/delx,dely,delt1,Bi,ti
        common /area2/tbc_l,tbc_r
        Up = sqrt(U*U+V*V)
        if (U.ge.0 .and. V.ge.0) index = 1
        if (U.lt.0 .and. V.ge.0) index = 2
        if (U.lt.0 .and. V.lt.0) index = 3
        if (U.ge.0 .and. V.lt.0) index = 4
        if(abs(U).lt.0.00001) then
          temp = pi/2.0
        else
          temp = atan(abs(V)/abs(U))
        endif
        go to (10,20,30,40),index
10      phi = temp
        go to 50
20      phi = pi - temp
        go to 50
30      phi = pi + temp
        go to 50
40      phi = 2*pi - temp
50      continue
        delzeta = Up * delt1
        delxq = delzeta * cos(pi+phi)
        delyq = delzeta * sin(pi+phi)
c  Temperature_q calculates the temperature of the point q which is upstream
```

```
c  of the point p i.e.(i,j).
      call temperature_q (i,j,delxq,delyq,tq,index)
      tp = tq
      if(delzeta.gt.delx .or. delzeta.gt.dely) then
        tp = (tbc_l+tbc_r)/2.0
        print *,'ERROR IN STEP1'
        print *,i,j,delzeta,tp
      endif
      return
      end

      subroutine step1b (U,i,j,tp)
      common /area1/delx,dely,delt1,Bi,ti
      common /area2/tbc_l,tbc_r
      common /area3/t1(50,10),tm(50,10),t2(50,10),sigma_t(50,10),Kh
      delzeta = U*delt1
      r = delzeta / delx
      if(U.ge.0.0) then
        temp = t1(i-1,j)
      else
        temp = t1(i+1,j)
      endif
      tp = r*temp + (1-r)*t1(i,j)
      if(delzeta.gt.delx) then
        tp = (tbc_l+tbc_r)/2.0
        print *,'ERROR IN STEP1B'
        print *,i,j,delzeta,tp
      endif
      return
      end

      subroutine temperature_q (i,j,delxq,delyq,tq,index)
      real n1,n2,n3,n4,neta
      common /area1/delx,dely,delt1,Bi,ti
      common /area3/t1(50,10),tm(50,10),t2(50,10),sigma_t(50,10),Kh
      xp = (i-1)*delx
      yp = (j-1)*dely
      xq = xp + delxq
      yq = yp + delyq
      ki = 1
      if(index.eq.1 .or. index.eq.4) ki = 0
      i1 = i-1+ki
      kj = 1
      if(index.eq.1 .or. index.eq.2) kj = 0
      j1 = j-1+kj
      i2 = i1+1
      j2 = j1
      i3 = i1+1
      j3 = j1+1
      i4 = i1
      j4 = j1+1
      x1 = (i1-1)*delx
      y1 = (j1-1)*dely
      neta = (2*(xq-x1)/delx) - 1
      zeta = (2*(yq-y1)/dely) - 1
      n1 = (1-neta)*(1-zeta)/4
      n2 = (1+neta)*(1-zeta)/4
      n3 = (1+neta)*(1+zeta)/4
      n4 = (1-neta)*(1+zeta)/4
      tq = n1*t1(i1,j1) + n2*t1(i2,j2) + n3*t1(i3,j3) + n4*t1(i4,j4)
      return
      end

      subroutine step2 (n,m)
      implicit real (k)
      real intertemp(50,10),ar(50),br(50),cr(50),fr(50),temp3(50)
```

```
      real arc(50)
      common /area1/delx,dely,delt1,Bi,ti
      common /area3/t1(50,10),tm(50,10),t2(50,10),sigma_t(50,10),Kh
      deltadi = delt1/2.0
      c1 = -2.0*dely*Bi
      c2 = -2.0*dely*Bi*ti
      call assign_bc(tm,n,m)
      do j = 1, m
        do i = 2 , n-1
          pe = sigma_t(i,j) / Kh
          temp1 = deltadi / (pe*delx*delx)
          temp2 = deltadi / (pe*dely*dely)
          a = -temp1
          bleft = 1 + 2.0*temp1
          bright = 1 - 2.0*temp2
          c = -temp1
          d = temp2
          e = temp2
          ar(i-1) = a
          br(i-1) = bleft
          cr(i-1) = c
          if(j.lt.m.and.j.gt.1) then
            temp5 = tm(i,j+1)
            temp4 = tm(i,j-1)
          endif
          if(j.eq.1) then
            temp5 = tm(i,j+1)
            temp4 = tm(i,j+1) + c1*tm(i,j) - c2
          endif
          if(j.eq.m) then
            temp5 = tm(i,j-1) + c1*tm(i,j) - c2
            temp4 = tm(i,j-1)
          endif
          fr(i-1) = d*temp5 + bright*tm(i,j) + e*temp4
        end do
        fr(1) = fr(1) - ar(1)*tm(1,j)
        fr(n-2) = fr(n-2) - cr(n-2)*tm(n,j)
        do i1 = 2,n-2
          arc(i1-1) = ar(i1)
        end do
        call tridia(arc,br,cr,fr,n-2,temp3)
        do i=1,n-2
          intertemp(i+1,j) = temp3(i)
        end do
      end do
      call assign_bc(intertemp,n,m)

      do i = 2, n-1
        do j = 1 , m
          pe = sigma_t(i,j) / Kh
          temp1 = deltadi / (pe*delx*delx)
          temp2 = deltadi / (pe*dely*dely)
          d = -temp2
          bleft = 1 + 2.0*temp2
          bright = 1 - 2.0*temp1
          e = -temp2
          a = temp1
          c = temp1
          fr(j) = c*intertemp(i+1,j) + a*intertemp(i-1,j) + bright*
     1    intertemp(i,j)
          if (j.ne.1 .and. j.ne.m) then
          ar(j-1) = e
          br(j) = bleft
          cr(j) = d
          endif
          if (j.eq.1) then
```

```
      br(1) = bleft + e*c1
      cr(1) = d + e
      fr(1) = fr(1) + e*c2
      endif
      if (j.eq.m) then
      br(m) = bleft + d*c1
      ar(m-1) = d + e
      fr(m) = fr(m) + d*c2
      endif
    end do
    call tridia(ar,br,cr,fr,m,temp3)
    do j=1,m
      t2(i,j) = temp3(j)
    end do
  end do
  call assign_bc(t2,n,m)

  return
  end

  subroutine assign_bc(t,n,m)
  dimension t(50,10)
  common /area2/tbc_l,tbc_r
  do 11 j=1,m
  t(1,j) = tbc_l
11 t(n,j) = tbc_r
  return
  end

  subroutine edgegrad (grad,p,i,j,n,m,index)
  dimension p(50,10)
  common /area1/delx,dely,delt1,Bi,ti
  if (index.eq.1) then
    p1 = p(1,j)
    p2 = p(2,j)
    h  = delx
    go to 10
  endif
  if (index.eq.2) then
    p1 = p(n,j)
    p2 = p(n-1,j)
    p3 = p(n-2,j)
    h  = -delx
  endif
  if (index.eq.3) then
    p1 = p(i,1)
    p2 = p(i,2)
    p3 = p(i,3)
    h  = dely
  endif
  if (index.eq.4) then
    p1 = p(i,m)
    p2 = p(i,m-1)
    p3 = p(i,m-2)
    h  = -dely
  endif
  grad = (4.0*p2 -p3 -3.0*p1) / (2 0*h)
  return
10 grad = (p2-p1)/h
  return
  end
```